Learning Deep Generative Models

Ruslan Salakhutdinov

Departments of Computer Science and Statistical Sciences, University of Toronto, Toronto M5S 3G4, Canada; email: rsalakhu@cs.toronto.edu

Keywords

deep learning, deep belief networks, deep Boltzmann machines, graphical models

Abstract

Building intelligent systems that are capable of extracting high-level representations from high-dimensional sensory data lies at the core of solving many artificial intelligence–related tasks, including object recognition, speech perception, and language understanding. Theoretical and biological arguments strongly suggest that building such systems requires models with deep architectures that involve many layers of nonlinear processing. In this article, we review several popular deep learning models, including deep belief networks and deep Boltzmann machines. We show that (*a*) these deep generative models, which contain many layers of latent variables and millions of parameters, can be learned efficiently, and (*b*) the learned high-level feature representations can be successfully applied in many application domains, including visual object recognition, information retrieval, classification, and regression tasks.

1. INTRODUCTION

Extraction of meaningful representations from rich sensory input lies at the core of solving many artificial intelligence (AI)-related tasks, including visual object recognition, speech perception, and language comprehension. Theoretical and biological arguments strongly suggest that building such systems requires deep architectures—models composed of several layers of nonlinear processing.

Many existing machine learning algorithms use what are called shallow architectures, including neural networks with only one hidden layer, kernel regression, and support vector machines, among many others. Theoretical results show that the internal representations learned by such systems are necessarily simple and are incapable of extracting certain types of complex structure from rich sensory input (Bengio & LeCun 2007, Bengio 2009). Training these systems also requires large amounts of labeled training data. By contrast, it appears that, for example, object recognition in the visual cortex uses many layers of nonlinear processing and requires very little labeled input (Lee et al. 1998). Thus, development of 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.

In general, models with deep architectures, including multilayer neural networks, are composed of several layers of parameterized nonlinear modules, so the associated loss functions are almost always nonconvex. The presence of many bad local optima or plateaus in the loss function makes deep models far more difficult to optimize in comparison with shallow models. Local gradient-based optimization algorithms, such as the backpropagation algorithm (Rumelhart et al. 1986), require careful parameter initialization and can often get trapped in a poor local optimum, particularly when training models with more than two or three layers (Sutskever et al. 2013). By contrast, models with shallow architectures (e.g., support vector machines) generally use convex loss functions, typically allowing one to carry out parameter optimization efficiently. The appeal of convexity has steered most machine learning research into developing learning algorithms that can be cast in terms of solving convex optimization problems.

Recently, Hinton et al. (2006) introduced a moderately fast, unsupervised learning algorithm for deep generative models called deep belief networks (DBNs). DBNs are probabilistic graphical models that contain multiple layers of hidden variables. Each nonlinear layer captures progressively more complex patterns of data, which represents a promising way of solving problems associated with visual object recognition, language comprehension, and speech perception (Bengio 2009). A key feature of the new learning algorithm for DBNs is its layer-by-layer training, which can be repeated several times to efficiently learn a deep, hierarchical probabilistic model. The new learning algorithm has excited many researchers in the machine learning community, primarily because of the following three crucial characteristics:

1. 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.
2. The learning algorithm can make efficient use of very large sets of unlabeled data, so the model can be pretrained in a completely unsupervised fashion. The very limited labeled data can then be used to only slightly fine-tune the model for a specific task at hand using standard gradient-based optimization.
3. There is an efficient way of performing approximate inference, which makes the values of the latent variables in the deepest layer easy to infer.

The strategy of layerwise unsupervised training followed by supervised fine-tuning allows efficient training of deep networks and gives promising results for many challenging learning

problems, substantially improving upon the current state of the art (Hinton et al. 2012, Krizhevsky et al. 2012). Many variants of this model have been successfully applied not only for classification tasks (Hinton et al. 2006, Bengio et al. 2007, Larochelle et al. 2009), but also for regression tasks (Salakhutdinov & Hinton 2008), visual object recognition (Krizhevsky et al. 2012, Lee et al. 2009, Ranzato et al. 2008), speech recognition (Hinton et al. 2012, Mohamed et al. 2012), dimensionality reduction (Hinton & Salakhutdinov 2006, Salakhutdinov & Hinton 2007), information retrieval (Torralba et al. 2008, Salakhutdinov & Hinton 2009c, Uria et al. 2014), natural language processing (Collobert & Weston 2008, Socher et al. 2011, Wang et al. 2012), extraction of optical flow information (Memisevic & Hinton 2010), prediction of quantitative structure–activity relationships (QSARs) (Dahl et al. 2014), and robotics (Lenz et al. 2013).

Another key advantage of these models is that they are able to capture nonlinear distributed representations. This is in sharp contrast to traditional probabilistic mixture-based latent variable models, including topic models (Hofmann 1999, Blei 2014) that are often used to analyze and extract semantic topics from large text collections. Many of the existing topic models, including a popular Bayesian admixture model, latent Dirichlet allocation (Blei et al. 2003), are based on the assumption that each document is represented as a mixture of topics, and each topic defines a probability distribution over words. All of these models can be viewed as graphical models in which latent topic variables have directed connections to observed variables that represent words in a document. One major drawback is that exact inference in these models is intractable, so one has to resort to slow or inaccurate approximations to compute the posterior distribution over topics. A second major drawback, which is shared by all mixture models, is that these models can never make predictions for words that are sharper than the distributions predicted by any of the individual topics. They are unable to capture the essential idea of distributed representations, namely that the distributions predicted by individual active features get multiplied together (and renormalized) to give the distribution predicted by a whole set of active features. This allows individual features to be fairly general but their intersection to be much more precise. For example, distributed representations allow the topics "government," "mafia," and "playboy" to combine to give very high probability to the word "Berlusconi," which is not predicted nearly as strongly by each topic alone. As shown by Welling et al. (2005) and Salakhutdinov & Hinton (2009b), models that use nonlinear distributed representations are able to generalize much better than latent Dirichlet allocation in terms of both the log-probability on previously unseen data vectors and the retrieval accuracy.

In this article, we provide a general overview of many popular deep learning models, including deep belief networks (DBNs) and deep Boltzmann machines (DBMs). In Section 2, we introduce restricted Boltzmann machines (RBMs), which form component modules of DBNs and DBMs, as well as their generalizations to exponential family models. In Section 3, we discuss DBNs and provide a thorough technical review of the greedy learning algorithm for DBNs. Section 4 focuses on new learning algorithms for a different type of hierarchical probabilistic model, the DBM. Finally, Section 5 presents a multimodal DBM that can extract a unified representation by learning a joint density model over the space of multimodal inputs (e.g., images and text, or video and sound).

2. RESTRICTED BOLTZMANN MACHINES AND THEIR GENERALIZATIONS

Restricted Boltzmann machines (RBMs) have been used effectively in modeling distributions over binary-valued data. Recent work on Boltzmann machines and their generalizations to exponential family distributions (Welling et al. 2005) have allowed these models to be successfully used in
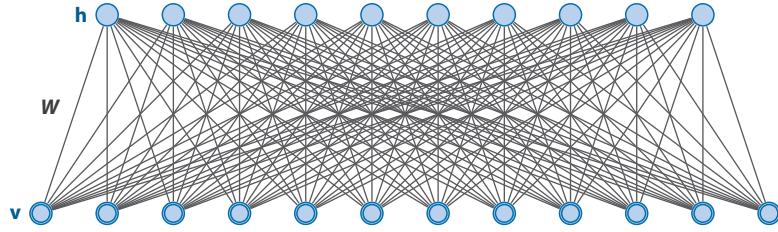
Figure 1

Restricted Boltzmann machine. The top layer represents a vector of "hidden" stochastic binary variables **h,** and the bottom layer represents a vector of "visible" stochastic binary variables **v**.

many application domains. In addition to reviewing standard RBMs, this section also reviews Gaussian–Bernoulli RBMs suitable for modeling real-valued inputs for image classification and speech recognition tasks (Lee et al. 2009, Taylor et al. 2010, Mohamed et al. 2012), as well as the replicated softmax model (Salakhutdinov & Hinton 2009b), which have been used for modeling sparse count data, such as word count vectors in a document. These models serve as our building blocks for other hierarchical models, including DBNs and DBMs.

2.1. Binary Restricted Boltzmann Machines

An RBM is a particular type of Markov random field that has a two-layer architecture (Smolensky 1986), in which the "visible" stochastic binary variables $\mathbf{v} \in \{0, 1\}^D$ are connected to "hidden" stochastic binary variables $\mathbf{h} \in \{0, 1\}^F$, as shown in **Figure 1**. The energy of the joint state $\{\mathbf{v}, \mathbf{h}\}$ is defined as follows:

$$
\begin{aligned}
E(\mathbf{v}, \mathbf{h}; \theta) &= -\mathbf{v}^\top W \mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{a}^\top \mathbf{h} \\
&= -\sum_{i=1}^{D} \sum_{j=1}^{F} W_{ij} v_i h_j - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{F} a_j h_j,
\end{aligned}
\tag{1}
$$

where $\theta = \{W, \mathbf{b}, \mathbf{a}\}$ are the model parameters. W_{ij} represents the symmetric interaction term between visible variable i and hidden variable j, and b_i and a_j are bias terms. The joint distribution over the visible and hidden variables is defined by

$$
P(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp(-E(\mathbf{v}, \mathbf{h}; \theta)),
\tag{2}
$$

$$
\mathcal{Z}(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}; \theta)).
\tag{3}
$$

$\mathcal{Z}(\theta)$ is known as the partition function or normalizing constant. The model then assigns the following probability to a visible vector $\mathbf{v}$:

$$
P(\mathbf{v}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right).
\tag{4}
$$

Because RBMs have a special bipartite structure, the hidden variables can be explicitly marginalized out, as follows:

$$
\begin{aligned}
P(\mathbf{v}; \theta) &= \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(\mathbf{v}^\top W \mathbf{h} + \mathbf{b}^\top \mathbf{v} + \mathbf{a}^\top \mathbf{h}\right) \\
&= \frac{1}{\mathcal{Z}(\theta)} \exp(\mathbf{b}^\top \mathbf{v}) \prod_{j=1}^{F} \sum_{b_j \in \{0,1\}} \exp\left(a_j b_j + \sum_{i=1}^{D} W_{ij} v_i b_j\right) \\
&= \frac{1}{\mathcal{Z}(\theta)} \exp(\mathbf{b}^\top \mathbf{v}) \prod_{j=1}^{F} \left(1 + \exp\left(a_j + \sum_{i=1}^{D} W_{ij} v_i\right)\right).
\end{aligned}
\tag{5}
$$

The conditional distributions over hidden variables $\mathbf{h}$ and visible vectors $\mathbf{v}$ can be easily derived from Equation 2 and are given by the following logistic functions:

$$
P(\mathbf{h}|\mathbf{v}; \theta) = \prod_{j} p(b_j|\mathbf{v}), \qquad P(\mathbf{v}|\mathbf{h}; \theta) = \prod_{i} p(v_i|\mathbf{h}),
\tag{6}
$$

$$
p(b_j = 1|\mathbf{v}) = g\left(\sum_{i} W_{ij} v_i + a_j\right),
\tag{7}
$$

$$
p(v_i = 1|\mathbf{h}) = g\left(\sum_{j} W_{ij} b_j + b_i\right),
\tag{8}
$$

where $g(x) = 1/(1 + \exp(-x))$ is the logistic function.

Given a set of observations $\{\mathbf{v}_n\}_{n=1}^{N}$, the derivative of the log-likelihood with respect to the model parameters W_{ij} is obtained from Equation 4:

$$
\begin{aligned}
\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log P(\mathbf{v_n}; \theta)}{\partial W_{ij}} &= \mathbb{E}_{P_{\text{data}}}[v_i b_j] - \mathbb{E}_{P_{\text{model}}}[v_i b_j], \\
\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log P(\mathbf{v_n}; \theta)}{\partial a_j} &= \mathbb{E}_{P_{\text{data}}}[b_j] - \mathbb{E}_{P_{\text{model}}}[b_j], \\
\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log P(\mathbf{v_n}; \theta)}{\partial b_i} &= \mathbb{E}_{P_{\text{data}}}[v_i] - \mathbb{E}_{P_{\text{model}}}[v_i],
\end{aligned}
\tag{9}
$$

where $\mathbb{E}_{P_{\text{data}}}[\cdot]$ denotes an expectation with respect to the data distribution $P_{\text{data}}(\mathbf{h}, \mathbf{v}; \theta) = P(\mathbf{h}|\mathbf{v}; \theta) P_{\text{data}}(\mathbf{v})$, where $P_{\text{data}}(\mathbf{v}) = \frac{1}{N} \sum_{n} \delta(\mathbf{v} - \mathbf{v}_n)$ represents the empirical distribution, and $\mathbb{E}_{P_{\text{model}}}[\cdot]$ is an expectation with respect to the distribution defined by the model, as in Equation 2. We sometimes refer to $\mathbb{E}_{P_{\text{data}}}[\cdot]$ as the data-dependent expectation and to $\mathbb{E}_{P_{\text{model}}}[\cdot]$ as the model's expectation.

Exact maximum likelihood learning in this model is intractable because exact computation of the expectation $\mathbb{E}_{P_{\text{model}}}[\cdot]$ takes time that is exponential in $\min\{D, F\}$, i.e., the number of visible or hidden variables. In practice, learning is done by following an approximation to the gradient of a different objective function, called the Contrastive Divergence (CD) algorithm (Hinton 2002):

$$
\Delta W = \alpha(\mathbb{E}_{P_{\text{data}}}[\mathbf{v}\mathbf{h}^\top] - \mathbb{E}_{P_T}[\mathbf{v}\mathbf{h}^\top]),
\tag{10}
$$

where α is the learning rate and P_T represents a distribution defined by running a Gibbs chain initialized at the data for T full steps. The special bipartite structure of RBMs allows for an efficient Gibbs sampler that alternates between sampling the states of the hidden variables

independently given the states of the visible variables and vice versa (see Equation 6). Setting $T = \infty$ recovers maximum likelihood learning. In many application domains, however, the CD learning with $T = 1$ (or CD1) has been shown to work quite well (Hinton 2002, Welling et al. 2005, Larochelle et al. 2009).

2.2. Gaussian–Bernoulli Restricted Boltzmann Machines

When modeling real-valued vectors, such as pixel intensities of image patches, one can easily extend RBMs to the Gaussian–Bernoulli variant (Hinton & Salakhutdinov 2006). In particular, consider modeling visible real-valued variables $\mathbf{v} \in \mathbb{R}^D$, and let $\mathbf{h} \in \{0, 1\}^F$ be stochastic binary hidden variables. The energy of the joint state $\{\mathbf{v}, \mathbf{h}\}$ of the Gaussian RBM is defined as follows:

$$E(\mathbf{v}, \mathbf{h}; \theta) = \sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} h_j \frac{v_i}{\sigma_i} - \sum_{j=1}^{F} a_j h_j, \tag{11}$$

where $\theta = \{W, \mathbf{a}, \mathbf{b}, \sigma^2\}$ are the model parameters.

The marginal distribution over the visible vector $\mathbf{v}$ takes the following form:

$$P(\mathbf{v}; \theta) = \sum_{\mathbf{h}} \frac{\exp(-E(\mathbf{v}, \mathbf{h}; \theta))}{\int_{\mathbf{v}'} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}; \theta)) d\mathbf{v}'}. \tag{12}$$

From Equation 11, derivation of the following conditional distributions is straightforward:

$$p(v_i = x | \mathbf{h}) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{\left(x - b_i - \sigma_i \sum_j h_j W_{ij}\right)^2}{2\sigma_i^2} \right), \tag{13}$$

$$p(h_j = 1 | \mathbf{v}) = g\left(b_j + \sum_i W_{ij} \frac{v_i}{\sigma_i} \right), \tag{14}$$

where $g(x) = 1/(1 + \exp(-x))$ is the logistic function. Observe that conditioned on the states of the hidden variables (Equation 13), each visible unit is modeled by a Gaussian distribution, the mean of which is shifted by the weighted combination of the hidden unit activations. The derivative of the log-likelihood with respect to W takes the following form:

$$\frac{\partial \log P(\mathbf{v}; \theta)}{\partial W_{ij}} = \mathbb{E}_{P_{\text{data}}} \left[\frac{1}{\sigma_i} v_i h_j \right] - \mathbb{E}_{P_{\text{model}}} \left[\frac{1}{\sigma_i} v_i h_j \right].$$

As discussed in Section 2.1, learning of the model parameters, including the variance σ^2, can be carried out using CD. In practice, however, instead of learning σ^2, one would typically use a fixed, predetermined value for σ^2 (Nair & Hinton 2009, Hinton & Salakhutdinov 2006).

Figure 2 shows a random subset of parameters W, also known as receptive fields, learned by a standard binary RBM and a Gaussian–Bernoulli RBM using Contrastive Divergence CD1. Observe that both RBMs learn highly localized receptive fields.

2.3. Replicated Softmax Model

The replicated softmax model represents another extension of the RBM and is used for modeling sparse count data, such as word count vectors in a document (Salakhutdinov & Hinton 2009b, 2013). Consider an undirected graphical model that consists of one visible layer and one hidden layer, as shown in **Figure 3**. This model is a type of RBM in which the visible variables that are usually binary have been replaced by softmax variables, each of which can have one of a number of different states.
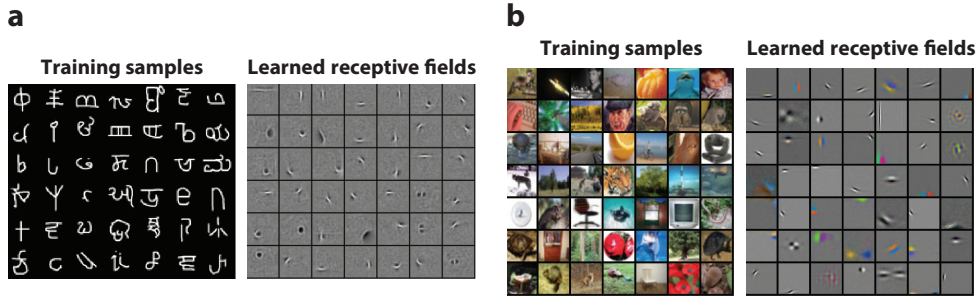
a

Training samples **Learned receptive fields**

b

Training samples **Learned receptive fields**

Figure 2

A random subset of the training images along with the learned receptive fields. (*a*) The binary restricted Boltzmann machine (RBM) trained on the Handwritten Characters data set (resolution is 28×28). (*b*) The Gaussian–Bernoulli RBM trained on the CIFAR-100 data set (resolution is 32×32). Each square displays the incoming weights from all of the visible variables into one hidden unit.

Specifically, let K be the dictionary size, M be the number of words appearing in a document, and $\mathbf{h} \in \{0, 1\}^F$ be stochastic binary hidden topic features. Let V be an $M \times K$ observed binary matrix with $v_{ik} = 1$ if visible unit i takes on value k (meaning the ith word in the document is the kth dictionary word). The energy of the state $\{V, \mathbf{h}\}$ can be defined as follows:

$$E(V, \mathbf{h}) = -\sum_{i=1}^{M} \sum_{j=1}^{F} \sum_{k=1}^{K} W_{ijk} h_j v_{ik} - \sum_{i=1}^{M} \sum_{k=1}^{K} v_{ik} b_{ik} - \sum_{j=1}^{F} h_j a_j, \tag{15}$$

where $\{W, \mathbf{a}, \mathbf{b}\}$ are the model parameters. W_{ijk} is a symmetric interaction term between visible variable i that takes on value k and hidden variable j, b_{ik} is the bias of unit i that takes on value k, and a_j is the bias of hidden feature j. The model assigns the following probability to a visible binary matrix V:

$$P(V; \theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp(-E(V, \mathbf{h}; \theta)), \quad \mathcal{Z}(\theta) = \sum_{V} \sum_{\mathbf{h}} \exp(-E(V, \mathbf{h}; \theta)). \tag{16}$$

Now suppose that for each document, we create a separate RBM with as many softmax units as there are words in the document. Assuming we can ignore the order of the words, all of these softmax units can share the same set of weights connecting them to binary hidden units. In this case, the energy of the state $\{V, \mathbf{h}\}$ for a document that contains M words is defined as follows:

$$E(V, \mathbf{h}) = -\sum_{j=1}^{F} \sum_{k=1}^{K} W_{jk} h_j \hat{v}_k - \sum_{k=1}^{K} \hat{v}_k b_k - M \sum_{j=1}^{F} h_j a_j, \tag{17}$$

a

Latent topics **h**

W_1 W_2

W_1 W_2

Observed softmax visibles **v**

W_1 W_2

W_1 W_1 W_2 W_2

b

Latent topics **h**

W_1 W_2

Multinomial visible **v**

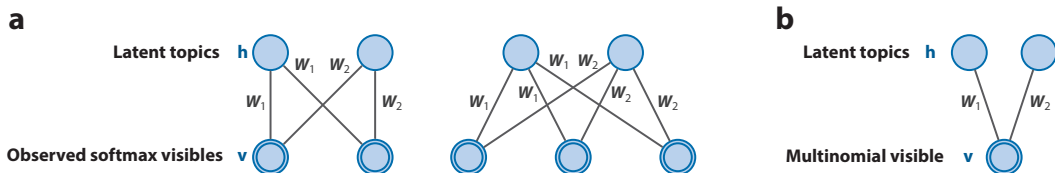**Figure 3**

The replicated softmax model. The top layer represents a vector **h** of stochastic binary hidden topic features, and the bottom layer consists of softmax visible variables, **v**. All visible variables share the same set of weights, connecting them to the binary hidden variables. (*a*) Two members of a replicated softmax family for documents containing two and three words. (*b*) A different interpretation of the replicated softmax model, in which M softmax variables with identical weights are replaced by a single multinomial variable that is sampled M times.

where $\hat{v}^k = \sum_{i=1}^{M} v_i^k$ denotes the count for the kth word. The bias terms of the hidden units are scaled up by the length of the document. This scaling is crucial and allows hidden units to behave sensibly when dealing with documents of different lengths.

The corresponding conditional distributions are given by the following equations:

$$p(b_j = 1|V) = g\left(M\,a_j + \sum_{k=1}^{K} \hat{v}_k W_{jk}\right), \tag{18}$$

$$p(v_{ik} = 1|\mathbf{h}) = \frac{\exp\left(b_k + \sum_{j=1}^{F} b_j W_{jk}\right)}{\sum_{q=1}^{K} \exp\left(b_q + \sum_{j=1}^{F} b_j W_{jq}\right)}. \tag{19}$$

We also note that using M softmax variables with identical weights is equivalent to having a single visible multinomial variable with support $\{1, \ldots, K\}$ that is sampled M times (see **Figure 3b**).

A pleasing property of using softmax variables is that the mathematics underlying the learning algorithm for binary RBMs remains the same. Given a collection of N documents $\{V_n\}_{n=1}^{N}$, the derivative of the log-likelihood with respect to parameters W takes the following form:

$$\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log P(\mathbf{V_n})}{\partial W_{jk}} = \mathbb{E}_{P_{\text{data}}}[\hat{v}_k b_j] - \mathbb{E}_{P_{\text{model}}}[\hat{v}_k b_j].$$

Similar to other types of RBMs, learning can be performed using CD.

Table 1 shows one-step reconstructions of some bags of words to illustrate what this replicated softmax model is learning (Srivastava & Salakhutdinov 2014). The model was trained using text from the MIR-Flickr data set (Huiskes & Lew 2008). The words in the left column were presented as input to the model, after which Equation 18 was used to compute a distribution over hidden units. Taking these probabilities as the states of the hidden units, Equation 19 was used to obtain a distribution over words. The right column shows the words with the highest probabilities in that distribution. Observe that the model has learned a reasonable notion of semantic similarity. For example, "chocolate, cake" generalizes to "sweets, desserts, food." Note that the model is able to capture some regularities about language, discover synonyms across multiple languages, and learn about geographical relationships.

3. DEEP BELIEF NETWORKS

A single layer of binary features is not the best way to capture the structure in high-dimensional input data. In this section, we describe an efficient way to learn additional layers of binary features using deep belief networks (DBNs).

Table 1 Some examples of one-step reconstruction from the replicated softmax model

| Input	Reconstruction
chocolate, cake	cake, chocolate, sweets, dessert, cupcake, food, sugar, cream, birthday
nyc	nyc, newyork, brooklyn, queens, gothamist, manhattan, subway, streetart
dog	dog, puppy, perro, dogs, pet, filmshots, tongue, pets, nose, animal
flower, high, 花	flower, 花, high, japan, sakura, 日本, blossom, tokyo, lily, cherry
girl, rain, station, norway	norway, station, rain, girl, oslo, train, umbrella, wet, railway, weather
fun, life, children	children, fun, life, kids, child, playing, boys, kid, play, love
forest, blur	forest, blur, woods, motion, trees, movement, path, trail, green, focus
españa, agua, granada	españa, agua, spain, granada, water, andalucía, naturaleza, galicia, nieve
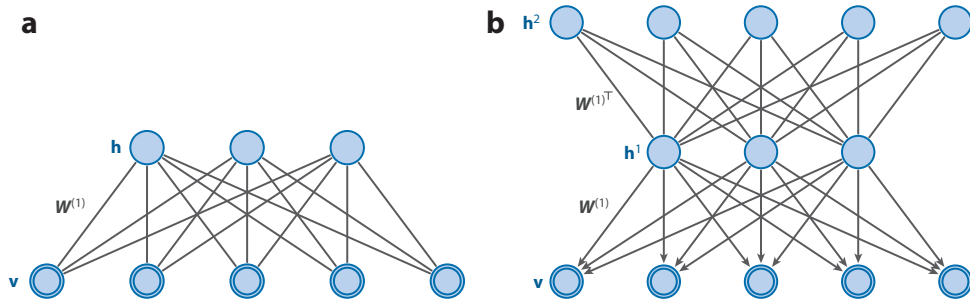
Figure 4

(*a*) A restricted Boltzmann machine (RBM). (*b*) A two-hidden-layer deep belief network (DBN) with tied weights $W^{(2)} = W^{(1)^\top}$. The joint distribution $P(\mathbf{v}, \mathbf{h}^{(1)}; W^{(1)})$ defined by this DBN is identical to the joint distribution $P(\mathbf{v}, \mathbf{h}^{(1)}; W^{(1)})$ defined by an RBM.

DBNs are probabilistic generative models that contain many layers of hidden variables, in which each layer captures high-order correlations between the activities of hidden features in the layer below. The top two layers of the DBN form an RBM model in which the lower layers form a directed sigmoid belief network, as shown in **Figure 4**. Hinton et al. (2006) introduced a fast unsupervised learning algorithm for these deep networks. A key feature of their algorithm is its greedy layer-by-layer training, which can be repeated several times to learn a deep hierarchical model. The learning procedure also provides an efficient way of performing approximate inference, which only requires a single bottom-up pass to infer the values of the top-level hidden variables.

Let us first consider learning a DBN with two layers of hidden variables $\{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}\}$. We also assume that the number of second-layer hidden variables is the same as the number of visible variables (see **Figure 4b**). The top two layers of the DBN form an undirected bipartite graph, an RBM, and the lower layers form a directed sigmoid belief network. The joint distribution over $\mathbf{v}$, $\mathbf{h}^{(1)}$, and $\mathbf{h}^{(2)}$ defined by this model takes the following form:[1]

$$P(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \theta) = P(\mathbf{v}|\mathbf{h}^{(1)}; W^{(1)})P(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}; W^{(2)}), \qquad (20)$$

where $\theta = \{W^{(1)}, W^{(2)}\}$ are the model parameters, $P(\mathbf{v}|\mathbf{h}^{(1)}; W^{(1)})$ is the directed sigmoid belief network, and $P(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}; W^{(2)})$ is the joint distribution defined by the second-layer RBM.

$$P(\mathbf{v}|\mathbf{h}^{(1)}; W^{(1)}) = \prod_i p(v_i|\mathbf{h}^{(1)}; W^{(1)}), \ p(v_i = 1|\mathbf{h}^{(1)}; W^{(1)}) = g\left(\sum_j W_{ij}^{(1)} b_j^{(1)}\right), \qquad (21)$$

$$P(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}; W^{(2)}) = \frac{1}{\mathcal{Z}(W^{(2)})} \exp(\mathbf{h}^{(1)^\top} W^{(2)}\mathbf{h}^{(2)}), \qquad (22)$$

where $g(x) = 1/(1 + \exp(-x))$ is the logistic function.

The greedy layer-by-layer learning strategy relies on the following key observation. Consider a two-hidden-layer DBN with tied parameters $W^{(2)} = W^{(1)^\top}$. The joint distribution of this DBN, $P(\mathbf{v}, \mathbf{h}^{(1)}; \theta) = \sum_{\mathbf{h}^{(2)}} P(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \theta)$, is identical to the joint distribution $P(\mathbf{v}, \mathbf{h}^{(1)}; W^{(1)})$ of the RBM (see Equation 2). Indeed, one can easily see from **Figure 4** that the marginal distribution over $\mathbf{h}^{(1)}$, $P(\mathbf{h}^{(1)}; W^{(1)})$ is the same for both models. Similarly, the conditional distribution $P(\mathbf{v}|\mathbf{h}^{(1)}; W^{(1)})$ is also the same for both models. To be more precise, using Equations 20–22 and the fact that

---

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

$W^{(2)} = W^{(1)^\top}$, we obtain the following joint distribution over $\{\mathbf{v}, \mathbf{h}^{(1)}\}$ of the DBN:

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

$$= \prod_i p(v_i|\mathbf{h}^{(1)}; W^{(1)}) \times \frac{1}{\mathcal{Z}(W^{(2)})} \prod_i \left(1 + \exp\left(\sum_j W_{ji}^{(2)} b_j^{(1)}\right)\right)$$

$$= \prod_i \frac{\exp\left(v_i \sum_j W_{ij}^{(1)} b_j^{(1)}\right)}{1 + \exp\left(\sum_j W_{ij}^{(1)} b_j^{(1)}\right)} \times \frac{1}{\mathcal{Z}(W^{(2)})} \prod_i \left(1 + \exp\left(\sum_j W_{ji}^{(2)} b_j^{(1)}\right)\right)$$

$$= \frac{1}{\mathcal{Z}(W^{(1)})} \prod_i \left(\exp\left(v_i \sum_j W_{ij}^{(1)} b_j^{(1)}\right)\right) \left[\text{because } W_{ji}^{(2)} = W_{ij}^{(1)}, \mathcal{Z}(W^{(1)}) = \mathcal{Z}(W^{(2)})\right]$$

$$= \frac{1}{\mathcal{Z}(W^{(1)})} \exp\left(\sum_{ij} W_{ij}^{(1)} v_i b_j^{(1)}\right), \tag{23}$$

which is identical to the joint distribution over $\{\mathbf{v}, \mathbf{h}^{(1)}\}$ defined by an RBM (Equation 2).

The greedy learning algorithm uses a stack of RBMs (see **Figure 5**) and proceeds as follows. We first train the bottom RBM with parameters $W^{(1)}$ as described in Section 2. We then initialize the second layer weights to $W^{(2)} = W^{(1)^\top}$, ensuring that the two-hidden-layer DBN is at least as good as our original RBM. We can now improve the fit of the DBN to the training data by untying and refitting parameters $W^{(2)}$.

For any approximating distribution $Q(\mathbf{h}^{(1)}|\mathbf{v})$ [provided that $Q(\mathbf{h}^{(1)}|\mathbf{v}) \neq 0$ whenever $P(\mathbf{v}, \mathbf{h}^{(1)}; \theta) \neq 0$], the log-likelihood of the two-hidden-layer DBN model has the following
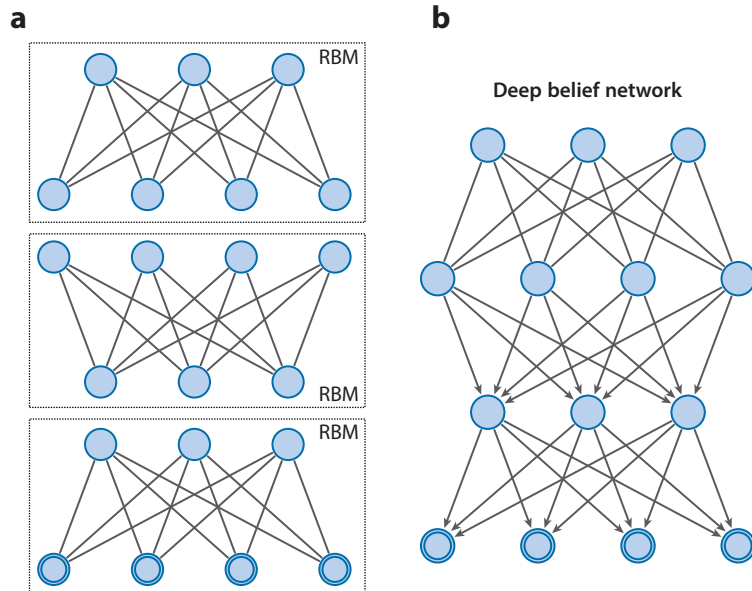

Figure 5

(*a*) Greedy learning of a stack of restricted Boltzmann machines (RBMs) in which the samples from the lower-level RBM are used as the data for training the next RBM. (*b*) The corresponding deep belief network.

variational lower bound, where the states $\mathbf{h}^{(2)}$ are analytically summed out:

$$
\begin{aligned}
\log P(\mathbf{v}; \theta) = \log \sum_{\mathbf{h}^{(1)}} P(\mathbf{v}, \mathbf{h}^{(1)}; \theta) &= \log \sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}) \frac{P(\mathbf{v}, \mathbf{h}^{(1)}; \theta)}{Q(\mathbf{h}^{(1)}|\mathbf{v})} \\
&\geq \sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}) \left[\log \frac{P(\mathbf{v}, \mathbf{h}^{(1)}; \theta)}{Q(\mathbf{h}^{(1)}|\mathbf{v})} \right] \text{(Jensen's inequality)} \\
&= \sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}) \left[\log P(\mathbf{v}, \mathbf{h}^{(1)}; \theta) \right] + \sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}) \left[\log \frac{1}{Q(\mathbf{h}^{(1)}|\mathbf{v})} \right] \\
&= \sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}) \left[\log P(\mathbf{h}^{(1)}; W^{(2)}) + \log P(\mathbf{v}|\mathbf{h}^{(1)}; W^{(1)}) \right] + H\left(Q(\mathbf{h}^{(1)}|\mathbf{v}) \right),
\end{aligned}
\tag{24}
$$

where $\mathcal{H}(\cdot)$ is the entropy functional. We set $Q(\mathbf{h}^{(1)}|\mathbf{v}) = P(\mathbf{h}^{(1)}|\mathbf{v}; W^{(1)})$, as defined by the bottom RBM (Equation 6). Initially, when $W^{(2)} = W^{(1)\top}$, Q is the true factorial posterior over $\mathbf{h}^{(1)}$ of the DBN, in which case the bound is tight. The strategy of the greedy learning algorithm is to fix the parameter vector $W^{(1)}$ and attempt to learn a better model for $P(\mathbf{h}^{(1)}; W^{(2)})$ by maximizing the variational lower bound of Equation 24 with respect to $W^{(1)}$. Maximizing this bound with fixed $W^{(1)}$ amounts to maximizing

$$
\sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}) \log P(\mathbf{h}^{(1)}; W^{(2)}),
\tag{25}
$$

which is equivalent to maximum likelihood training of the second-layer RBM with vectors $\mathbf{h}^{(1)}$ drawn from $Q(\mathbf{h}^{(1)}|\mathbf{v})$ as data. When presented with a data set of N training input vectors, the second-layer RBM, $P(\mathbf{h}^{(1)}; W^{(2)})$, will learn a better model of the aggregated posterior over $\mathbf{h}^{(1)}$, which is simply the mixture of factorial posteriors for all the training cases, $\frac{1}{N} \sum_n P(\mathbf{h}^{(1)}|\mathbf{v}_n; W^{(1)})$. Note that any increase in the variational lower bound that results from changing $W^{(2)}$ will also result in an increase of the data likelihood of the DBN.[2]

Algorithm 1 (Recursive greedy learning procedure for the deep belief network):

1: Fit the parameters $W^{(1)}$ of the first-layer RBM to data.
2: Fix the parameter vector $W^{(1)}$, and use samples $\mathbf{h}^{(1)}$ from $Q(\mathbf{h}^{(1)}|\mathbf{v}) = P(\mathbf{h}^{(1)}|\mathbf{v}, W^{(1)})$ as the data for training the next layer of binary features with an RBM.
3: Fix the parameters $W^{(2)}$ that define the second layer of features, and use the samples $\mathbf{h}^{(2)}$ from $Q(\mathbf{h}^{(2)}|\mathbf{h}^{(1)}) = P(\mathbf{h}^{(2)}|\mathbf{h}^{(1)}, W^{(2)})$ as the data for training the third layer of binary features.
4: Proceed recursively for the next layers.

This idea can be extended to training the third-layer RBM on vectors $\mathbf{h}^{(2)}$ drawn from the second-layer RBM. By initializing $W^{(3)} = W^{(2)\top}$, we are guaranteed to improve the lower bound on the log-likelihood, although changing $W^{(3)}$ to improve the bound can decrease the actual log-likelihood. This greedy layer-by-layer training can be repeated several times to learn a deep hierarchical model. The procedure is summarized in Algorithm 1.

---

[2]Improving the variational bound by changing $W^{(2)}$ will increase the log-likelihood because the bound is initially tight. When learning deeper layers, the variational bound is not initially tight, so even the initial improvement in the bound is not guaranteed to increase the log-likelihood.
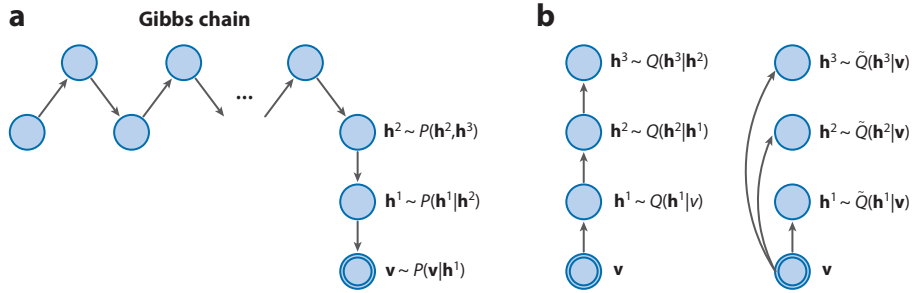
Figure 6

(*a*) Sample generation from the deep belief network. (*b*) Sample generation from approximate posterior $Q(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}|\mathbf{v})$ (*left*) versus from fully factorized approximate posterior $\tilde{Q}(\mathbf{h}^{(1)}|\mathbf{v})\tilde{Q}(\mathbf{h}^{(2)}|\mathbf{v})\tilde{Q}(\mathbf{h}^{(3)}|\mathbf{v})$ (*right*).

Algorithm 2 (Modified recursive greedy learning procedure for the deep belief network):

1: Fit the parameters $W^{(1)}$ of the first-layer RBM to data.
2: Fix the parameter vector $W^{(1)}$, and use samples $\mathbf{h}^{(1)}$ from $\tilde{Q}(\mathbf{h}^{(1)}|\mathbf{v}) = P(\mathbf{h}^{(1)}|\mathbf{v}, W^{(1)})$ as the data for training the next layer of binary features with an RBM.
3: Fix the parameters $W^{(2)}$ that define the second layer of features, and use the samples $\mathbf{h}^{(2)}$ from $\tilde{Q}(\mathbf{h}^{(2)}|\mathbf{v})$ as the data for training the third layer of binary features.
4: Proceed recursively for the next layers.

After training a DBN with L layers, the joint distribution of the model P and its approximate posterior distribution Q are given by:

$$P(\mathbf{v}, \mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}) = P(\mathbf{v}|\mathbf{h}^{(1)}) \ldots P(\mathbf{h}^{(L-2)}|\mathbf{h}^{(L-1)})P(\mathbf{h}^{(L-1)}, \mathbf{h}^{(L)}),$$

$$Q(\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}|\mathbf{v}) = Q(\mathbf{h}^{(1)}|\mathbf{v})Q(\mathbf{h}^{(2)}|\mathbf{h}^{(1)}) \ldots Q(\mathbf{h}^{(L)}|\mathbf{h}^{(L-1)}).$$

To generate an approximate sample from the DBN, we can run an alternating Gibbs sampler (Equation 6) to generate an approximate sample $\mathbf{h}^{(L-1)}$ from $P(\mathbf{h}^{(L-1)}, \mathbf{h}^{(L)})$, defined by the top-level RBM, followed by a top-down pass through the sigmoid belief network by stochastically activating each lower layer in turn (see **Figure 6*a***). To get a sample from the approximate posterior distribution Q, we simply perform a bottom-up pass by stochastically activating each higher layer in turn. The marginal distribution of the top-level hidden variables of our approximate posterior $Q(\mathbf{h}^{(L)}|\mathbf{v})$ will be nonfactorial and, in general, could be multimodal. For many practical applications (e.g., information retrieval), having an explicit form for $Q(\mathbf{h}^{(L)}|\mathbf{v})$, which allows efficient approximate inference, is of crucial importance. One possible alternative is to choose the following fully factorized approximating distribution $\tilde{Q}$:

$$\tilde{Q}(\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}|\mathbf{v}) = \prod_{l=1}^{L} \tilde{Q}(\mathbf{h}^{(l)}|\mathbf{v}), \tag{26}$$

where we define

$$\tilde{Q}(\mathbf{h}^{(1)}|\mathbf{v}) = \prod_{j} q(b_j^{(1)}|\mathbf{v}), \quad q(b_j^{(1)} = 1|\mathbf{v}) = g\left(\sum_i W_{ij}^{(1)} v_i + a_j^{(1)}\right), \text{ and} \tag{27}$$

$$\tilde{Q}(\mathbf{h}^{(l)}|\mathbf{v}) = \prod_{j} q(b_j^{(l)}|\mathbf{v}), \quad q(b_j^{(l)} = 1|\mathbf{v}) = g\left(\sum_i W_{ij}^{(l)} q(b_i^{(l-1)} = 1|\mathbf{v}) + a_j^{(l)}\right), \tag{28}$$

where $g(x) = 1/(1 + \exp(-x))$ and $l = 2, \ldots, L$. The factorial posterior $\tilde{Q}(\mathbf{h}^{(L)}|\mathbf{v})$ is obtained by simply replacing the stochastic hidden variables with real-valued probabilities and then performing a single deterministic bottom-up pass to compute $q(b_j^{(L)} = 1|\mathbf{v})$. This fully factorized approximation also suggests a modified greedy learning algorithm, summarized in Algorithm 2. In this modified algorithm, the samples used for training higher-level RBMs are instead taken from a fully factorized approximate posterior $\tilde{Q}$. Note that the modified algorithm does not guarantee to improve the lower bound on the log-probability of the training data. Nonetheless, this is the actual algorithm commonly used in practice (Hinton & Salakhutdinov 2006, Taylor et al. 2006, Torralba et al. 2008, Bengio 2009). The modified algorithm works well, particularly when a fully factorized $\tilde{Q}$ is used to perform approximate inference in the final model.

DBNs can also be used for classification and regression tasks. Many of the resulting extensions exploit the following two key properties of DBNs. First, they can be learned efficiently from large amounts of unlabeled data. Second, they can be discriminatively fine-tuned using the standard backpropagation algorithm. For example, a DBN can be used to extract useful feature representations that allow one to learn a good covariance kernel for a Gaussian process model (Salakhutdinov & Hinton 2008). The greedy learning algorithm can also be used to make nonlinear autoencoders work considerably better than widely used methods, such as principal component analysis (PCA) and singular value decomposition (SVD) (Hinton & Salakhutdinov 2006). Similarly, layer-by-layer pretraining followed by discriminative fine-tuning achieves good performance on phone recognition tasks, as well as on various audio classification tasks (Lee et al. 2009, Taylor et al. 2010, Mohamed et al. 2012).

4. DEEP BOLTZMANN MACHINES

In this section we present a new learning algorithm for a different type of hierarchical probabilistic model, called a deep Boltzmann machine (DBM). Unlike DBNs, a DBM is a type of Markov random field, or undirected graphical model, in which all connections between layers are undirected. DBMs are interesting for several reasons. First, similar to DBNs, DBMs have the ability to learn internal representations that capture 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, Mohamed et al. 2012). High-level representations are 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, if DBMs are learned in the right way there is a very fast way to initialize the states of the variables in all layers by a simple bottom-up pass. Third, unlike DBNs and many other models with deep architectures (Ranzato et al. 2007, Vincent et al. 2008, Serre et al. 2007), the approximate inference procedure, after the initial bottom-up pass, can incorporate top-down feedback, allowing DBMs to use higher-level knowledge to resolve uncertainty about intermediate-level features, thus creating better data-dependent representations, as well as better data-dependent statistics for learning.

Consider a three-hidden-layer DBM, as shown in **Figure 7b**, with no within-layer connections. The energy of the state $\{\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}\}$ is defined as

$$E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \theta) = -\mathbf{v}^\top W^{(1)}\mathbf{h}^{(1)} - \mathbf{h}^{(1)\top} W^{(2)}\mathbf{h}^{(2)} - \mathbf{h}^{(2)\top} W^{(3)}\mathbf{h}^{(3)}, \qquad (29)$$

where $\theta = \{W^{(1)}, W^{(2)}, W^{(3)}\}$ are the model parameters, representing visible-to-hidden and hidden-to-hidden symmetric interaction terms. The model assigns the following probability to a
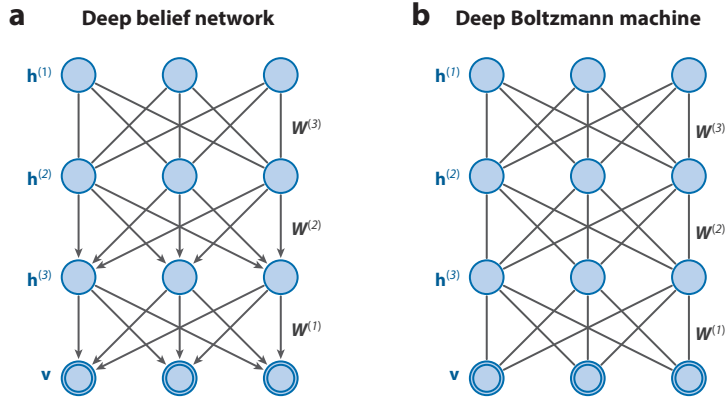
a   **Deep belief network**   **b**   **Deep Boltzmann machine**

Figure 7

(*a*) Deep belief network (DBN) in which the top two layers form an undirected graph and the remaining layers form a belief net with directed, top-down connections. (*b*) Deep Boltzmann machine (DBM) with visible-to-hidden and hidden-to-hidden connections, but no within-layer connections. All of the connections in a DBM are undirected.

visible vector $\mathbf{v}$:

$$P(\mathbf{v};\theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)}} \exp(-E(\mathbf{v},\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)};\theta)). \tag{30}$$

Observe that setting both $W^{(2)} = 0$ and $W^{(3)} = 0$ recovers the simpler RBM model. The conditional distributions over the visible and the three sets of hidden variables are given by the following logistic functions:

$$p(b_j^{(1)} = 1|\mathbf{v},\mathbf{h}^{(2)}) = g\left(\sum_i W_{ij}^{(1)} v_i + \sum_m W_{jm}^{(2)} b_m^{(2)}\right), \tag{31}$$

$$p(b_m^{(2)} = 1|\mathbf{h}^{(1)},\mathbf{h}^{(3)}) = g\left(\sum_j W_{jm}^{(2)} b_j^{(1)} + \sum_l W_{ml}^{(3)} b_l^{(3)}\right), \tag{32}$$

$$p(b_l^{(3)} = 1|\mathbf{h}^{(2)}) = g\left(\sum_m W_{ml}^{(3)} b_m^{(2)}\right), \tag{33}$$

$$p(v_i = 1|\mathbf{h}^{(1)}) = g\left(\sum_j W_{ij}^{(1)} b_j^{(1)}\right). \tag{34}$$

The derivative of the log-likelihood with respect to the model parameters takes the following form:

$$\frac{\partial \log P(\mathbf{v};\theta)}{\partial W^{(1)}} = \mathbb{E}_{P_{\text{data}}}[\mathbf{v}\mathbf{h}^{(1)\top}] - \mathbb{E}_{P_{\text{model}}}[\mathbf{v}\mathbf{h}^{(1)\top}], \tag{35}$$

where $\mathbb{E}_{P_{\text{model}}}[\cdot]$ denotes an expectation with respect to the distribution defined by the model, and $\mathbb{E}_{P_{\text{data}}}[\cdot]$ is an expectation with respect to the completed data distribution $P_{\text{data}}(\mathbf{h},\mathbf{v};\theta) = P(\mathbf{h}|\mathbf{v};\theta)P_{\text{data}}(\mathbf{v})$, where $P_{\text{data}}(\mathbf{v}) = \frac{1}{N}\sum_n \delta(\mathbf{v} - \mathbf{v}_n)$ represents the empirical distribution. The derivatives with respect to parameters $W^{(2)}$ and $W^{(3)}$ take similar forms, but they involve the outer products $\mathbf{h}^{(1)}\mathbf{h}^{(2)\top}$ and $\mathbf{h}^{(2)}\mathbf{h}^{(3)\top}$, respectively. Unlike RBMs, the conditional distribution over

the states of the hidden variables conditioned on the data is no longer factorial. The exact computation of the data-dependent expectation takes time that is exponential in the number of hidden variables, whereas the exact computation of the model's expectation takes time that is exponential in the number of hidden and visible variables.

4.1. Approximate Maximum Likelihood Learning

The original learning algorithm for general Boltzmann machines used randomly initialized Markov chains to approximate both of the expectations needed to approximate gradients of the likelihood function (Hinton & Sejnowski 1983). This learning procedure is too slow to be practical, however, so we now consider a variational approach to alleviate this problem. In the variational approach, mean-field inference is used to estimate data-dependent expectations, and a Markov chain Monte Carlo (MCMC)-based stochastic approximation procedure is used to approximate the model's expected sufficient statistics (Tieleman 2008, Salakhutdinov 2008, Salakhutdinov & Hinton 2009a).

Consider any approximating distribution $Q(\mathbf{h}|\mathbf{v}; \boldsymbol{\mu})$ for the posterior $P(\mathbf{h}|\mathbf{v}; \theta)$. Similar to the DBN bound given in Equation 24, the DBM's variational lower bound on the log-likelihood takes the following form:

$$\log P(\mathbf{v}; \theta) \geq \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{v}; \boldsymbol{\mu}) \log P(\mathbf{v}, \mathbf{h}; \theta) + \mathcal{H}(Q), \tag{36}$$

where $\mathcal{H}(\cdot)$ is the entropy functional. The bound becomes tight if and only if $Q(\mathbf{h}|\mathbf{v}; \boldsymbol{\mu}) = P(\mathbf{h}|\mathbf{v}; \theta)$.

For simplicity and speed, we approximate the true posterior using a fully factorized distribution (i.e., the naive mean-field approximation) over the three sets of hidden variables:

$$Q^{\mathrm{MF}}(\mathbf{h}|\mathbf{v}; \boldsymbol{\mu}) = \prod_{j=1}^{F_1} \prod_{k=1}^{F_2} \prod_{m=1}^{F_3} q(b_j^{(1)}) q(b_k^{(2)}) q(b_m^{(3)}), \tag{37}$$

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 of Equation 36 takes a particularly simple form:

$$\log P(\mathbf{v}; \theta) \geq \mathbf{v}^\top W^{(1)} \boldsymbol{\mu}^{(1)} + \boldsymbol{\mu}^{(1)\top} W^{(2)} \boldsymbol{\mu}^{(2)} + \boldsymbol{\mu}^{(2)\top} W^{(3)} \boldsymbol{\mu}^{(3)} \\ - \log \mathcal{Z}(\theta) + \mathcal{H}(Q). \tag{38}$$

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:

$$\mu_j^{(1)} \leftarrow \sigma \left(\sum_{i=1}^{D} W_{ij}^{(1)} v_i + \sum_{k=1}^{F_2} W_{jk}^{(2)} \mu_k^{(2)} \right), \tag{39}$$

$$\mu_k^{(2)} \leftarrow \sigma \left(\sum_{j=1}^{F_1} W_{jk}^{(2)} \mu_j^{(1)} + \sum_{m=1}^{F_3} W_{km}^{(3)} \mu_m^{(3)} \right), \tag{40}$$

$$\mu_m^{(3)} \leftarrow \sigma \left(\sum_{k=1}^{F_2} W_{km}^{(3)} \mu_k^{(2)} \right). \tag{41}$$

Note the close connection between the form of the mean-field fixed-point updates and the form of the conditional distributions defined by Equations 31–33.[3] To solve these fixed-point

---

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

equations, we simply cycle through layers, updating the mean-field parameters within a single layer. The variational parameters $\boldsymbol{\mu}$ are then used to compute the data-dependent statistics in Equation 35. For example,

$$\mathbb{E}_{P_{\text{data}}}[\mathbf{v}\mathbf{h}^{(1)\top}] = \frac{1}{N}\sum_{n=1}^{N}\mathbf{v}_n\mu_n^{(1)\top}$$

$$\mathbb{E}_{P_{\text{data}}}[\mathbf{h}^{(1)}\mathbf{h}^{(2)\top}] = \frac{1}{N}\sum_{n=1}^{N}\mu_n^{(1)}\mu_n^{(2)\top},$$

where the averages are taken over the training cases.

Given the variational parameters $\boldsymbol{\mu}$, the model parameters θ are then updated to maximize the variational bound using an MCMC-based stochastic approximation (Younes 2000, Tieleman 2008, Salakhutdinov & Hinton 2009a). In particular, let θ_t and $\mathbf{x}_t = \{\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}\}$ be the current parameters and the state, respectively. Then $\mathbf{x}_t$ and θ_t are updated sequentially as follows:

1. Given $\mathbf{x}_t$, sample a new state $\mathbf{x}_{t+1}$ from the transition operator $T_{\theta_t}(\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t)$ that leaves $P(\cdot; \theta_t)$ invariant. Sampling this new state is accomplished using Gibbs sampling (see Equations 31–33).
2. A new parameter θ_{t+1} is then obtained by making a gradient step, in which the intractable model's expectation $\mathbb{E}_{P_{\text{model}}}[\cdot]$ in the gradient is replaced by a point estimate at sample $\mathbf{x}_{t+1}$.

In practice, to reduce the variance of the estimator, we typically maintain a set of S Markov chains $X_t = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,S}\}$ and use an average over those particles.

Stochastic approximation provides asymptotic convergence guarantees and belongs to the general class of Robbins–Monro approximation algorithms (Robbins & Monro 1951, Younes 2000). Sufficient conditions that ensure almost sure convergence to an asymptotically stable point are given in articles by Younes (1989, 2000) and Yuille (2004). One necessary condition requires the learning rate to decrease with time, such that $\sum_{t=0}^{\infty}\alpha_t = \infty$ and $\sum_{t=0}^{\infty}\alpha_t^2 < \infty$. For example, this condition can be satisfied by $\alpha_t = a/(b + t)$, for positive constants $a > 0$, $b > 0$. In practice, the sequence $|\theta_t|$ is typically bounded, and the Markov chain, governed by the transition kernel T_θ, is typically ergodic. Together with the condition on the learning rate, these conditions are sufficient to ensure almost sure convergence of the stochastic approximation algorithm to an asymptotically stable point (Younes 2000, Yuille 2004).

The learning procedure for DBMs described above can be used by initializing model parameters at random. The model performs much better if parameters are initialized sensibly, however, so it is common to use a greedy layerwise pretraining strategy by learning a stack of RBMs (for details, see Salakhutdinov & Hinton 2009a). The pretraining procedure is quite similar to the one used for DBNs discussed in Section 3, and it allows us to perform approximate inference using a single bottom-up pass. This fast approximate inference can also be used to initialize the mean-field, which then converges much faster than does a mean-field initialized at random.

4.2. Evaluating Deep Boltzmann Machines as Discriminative Models

After learning, the stochastic activities of the binary features in each layer are replaced by deterministic, real-valued probabilities, and a DBM 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 $q(b_j^{(2)} = 1|\mathbf{v})$ of this approximate posterior, together with the data, are used to create what we call an augmented input for this deep multilayer neural network, as shown in **Figure 8**. This augmented input is
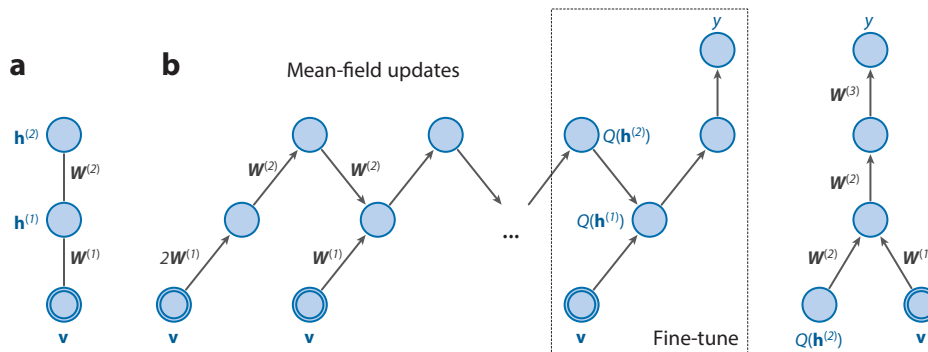
Figure 8

(*a*) A two-hidden-layer Boltzmann machine. (*b*) After learning, the deep Boltzmann machine is used to initialize a multilayer neural network. The marginals of the approximate posterior $q(b_j^{(2)} = 1|\mathbf{v})$ are used as additional inputs. The network is fine-tuned by backpropagation.

important because it maintains the scale of the inputs that each hidden variable expects. For example, the conditional distribution over $\mathbf{h}^{(1)}$, as defined by the DBM model (see Equation 31), takes the following form:

$$p(b_j^{(1)} = 1|\mathbf{v}, \mathbf{h}^{(2)}) = g\left(\sum_i W_{ij}^{(1)} v_i + \sum_m W_{jm}^{(2)} b_m^{(2)}\right).$$

Hence, the layer $\mathbf{h}^{(1)}$ receives inputs from both $\mathbf{v}$ and $\mathbf{h}^{(2)}$. When this DBM is used to initialize a feed-forward network (**Figure 8***b*), the augmented inputs $Q(\mathbf{h}^{(2)}|\mathbf{v})$ serve as a proxy for $\mathbf{h}^{(2)}$, ensuring that when the feed-forward network is fine-tuned using standard backpropagation of error derivatives, the hidden variables in $\mathbf{h}^{(1)}$ initially receive the same input as they would have received in a mean-field update during the pretraining stage.

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, the fine-tuning may drive the first-layer connections $W^{(2)}$ to zero, resulting 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 making its predictions on the basis of only the approximate posterior. However, the network typically makes use of the entire augmented input for making predictions.

4.3. Experimental Results

To illustrate the kinds of probabilistic models DBMs are capable of learning, we conducted experiments on two well-studied data sets: the MNIST and NORB data sets.

4.3.1. MNIST data set. The Mixed National Institute of Standards and Technology (MNIST) database of handwritten digits contains 60,000 training images and 10,000 test images of ten handwritten digits (0 to 9), each of which is centered within a 28×28 pixel image. Intermediate intensities between 0 and 255 were treated as probabilities, and we sampled binary values from these probabilities independently for each pixel each time an image was used. In our first experiment, we trained two DBMs: One had two hidden layers (500 and 1,000 hidden units), and the other had three (500, 500, and 1,000 hidden units), as shown in **Figure 9**. To estimate the model's partition function, we used annealed importance sampling (Neal 2001, Salakhutdinov
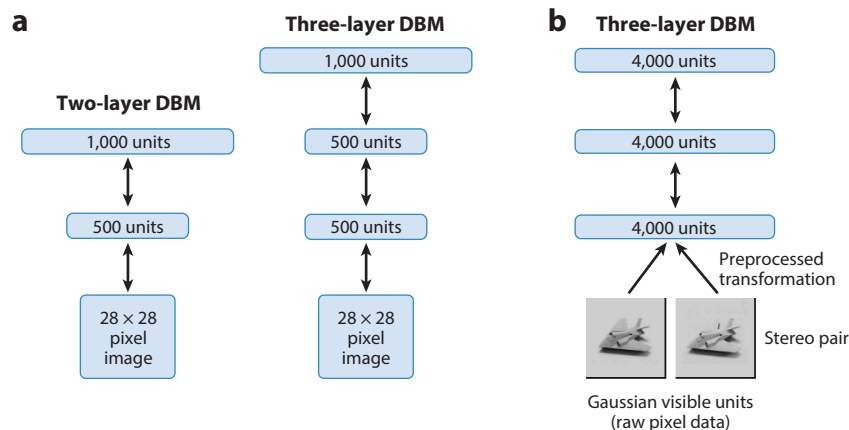
Figure 9

(*a*) The architectures of two deep Boltzmann machines (DBMs) used in Mixed National Institute of Standards and Technology (MNIST) experiments. (*b*) The architecture of a DBM used in New York University Object Recognition Benchmark (NORB) experiments.

& Murray 2008). The estimates of the variational lower bound (see Equation 36) on the average test log-probability were −84.62 and −85.10 for the two- and three-layer DBMs, respectively. Observe that even though the two DBMs contain about 0.9 and 1.15 million parameters, respectively, they do not appear to suffer much from overfitting. The difference between the estimates of the training and test log-probabilities was approximately 1 nat. **Figure 10** further shows samples generated from the two models by randomly initializing all binary states and running the Gibbs sampler for 100,000 steps. All samples look like the real handwritten digits.

For a simple comparison, we also trained several mixture-of-Bernoullis models. We used 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 −135.78. Compared with a DBM, a mixture-of-Bernoullis model performs very poorly. The difference of about 50 nats per test case is striking.
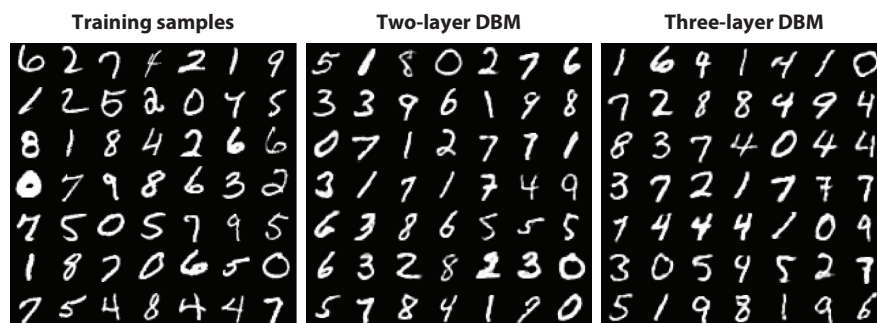
Figure 10

Random samples from the training set, and samples generated from two deep Boltzmann machines (DBMs) by running the Gibbs sampler for 100,000 steps. The images shown are the probabilities of the binary visible variables given the binary states of the hidden variables.

Finally, after discriminative fine-tuning, the two-hidden-layer DBM achieves an error rate of 0.95% on the full MNIST test set.[4] The three-layer DBM gives a slightly worse error rate of 1.01%. These DBM error rates are compared with error rates of 1.4%, achieved by support vector machines (SVMs) (Decoste & Schölkopf 2002); 1.6%, achieved by a randomly initialized backpropagation algorithm; 1.2%, achieved by the DBN described in articles by Hinton & Salakhutdinov (2006) and Hinton et al. (2006); and 0.97%, obtained by using a combination of discriminative and generative fine-tuning on the same DBN (Hinton 2007).

4.3.2. NORB data set. Results on the MNIST data set show that DBMs 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 the New York University Object Recognition Benchmark (NORB) data set, which is a considerably more difficult data set than the MNIST data set is. NORB (LeCun et al. 2004) contains images of 50 different three-dimensional (3D) toy objects, with 10 objects in each of 5 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, and 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.

We trained a two-hidden-layer DBM in which each layer contained 4,000 hidden variables, as shown in **Figure 9b**. 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 error rate is compared with rates of 11.6%, achieved by SVMs (Bengio & LeCun 2007); 22.5%, achieved by logistic regression; and 18.4%, achieved by the k-nearest-neighbor approach (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 a total of 1,166,400 training instances.[5] 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 11** shows samples generated from the model by running prolonged Gibbs sampling. Note that the model was able to capture many regularities in this high-dimensional, richly structured data, including variation in object classes, viewpoints, and lighting conditions.

Surprisingly, even though the DBM contains approximately 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 only to slightly adjust the layers of features already discovered by the DBM.

5. EXTENSIONS TO LEARNING FROM MULTIMODAL DATA

DBMs can be easily extended to modeling data that contain multiple modalities. The key idea is to learn a joint density model over the space of the multimodal inputs. For example, using a large collection of user-tagged images, we can learn a joint distribution over images and text, $P(\mathbf{v}_{\text{img}}, \mathbf{v}_{\text{txt}}; \theta)$ (Srivastava & Salakhutdinov 2014). By drawing samples from $P(\mathbf{v}_{\text{txt}}|\mathbf{v}_{\text{img}}; \theta)$ and from $P(\mathbf{v}_{\text{img}}, \mathbf{v}_{\text{txt}}; \theta)$, we can fill in missing data, thereby doing image annotation [for $P(\mathbf{v}_{\text{txt}}|\mathbf{v}_{\text{img}}; \theta)$] and image retrieval [for $P(\mathbf{v}_{\text{img}}, \mathbf{v}_{\text{txt}}; \theta)$].

---

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

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

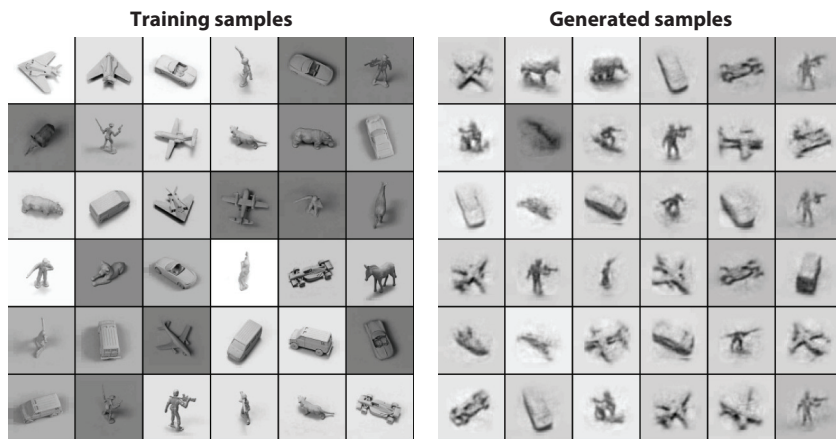**Training samples** **Generated samples**

Figure 11

Random samples from the training set (*left*), and samples generated from a three-hidden-layer deep Boltzmann machine by running the Gibbs sampler for 10,000 steps (*right*).

Let us construct a multimodal DBM using an image–text bimodal DBM as our running example. Let $\mathbf{v}^m \in \mathbb{R}^D$ denote a real-valued image input and $\mathbf{v}^t \in \{1, \ldots, K\}$ denote an associated text input containing M words, where v_k^t denotes the count for the kth word. We model each data modality using a separate two-layer DBM. We use a Gaussian–Bernoulli RBM as a first-layer model for the image-specific DBM (see **Figure 12a**). Hence, the image-specific DBM uses a Gaussian distribution to model the distribution over real-valued image features. Similarly, a text-specific DBM uses a replicated softmax model to model the distribution over word count vectors.

To form a multimodal DBM, we combine the two models by adding an additional layer on top of them. The resulting graphical model is shown in **Figure 12c**. Let $\mathbf{h}^{(1m)} \in \{0, 1\}^{F_1^m}$ and $\mathbf{h}^{(2m)} \in \{0, 1\}^{F_2^m}$ denote the two layers of hidden variables of the image-specific DBM, and let $\mathbf{h}^{(1t)} \in \{0, 1\}^{F_1^t}, \mathbf{h}^{(2t)} \in \{0, 1\}^{F_2^t}$ represent the two layers of hidden variables of the text-specific DBM. Then the joint distribution over the multimodal input, where $\mathbf{h} = \{\mathbf{h}^{(1m)}, \mathbf{h}^{(2m)}, \mathbf{h}^{(1t)}, \mathbf{h}^{(2t)}, \mathbf{h}^{(3)}\}$ denotes
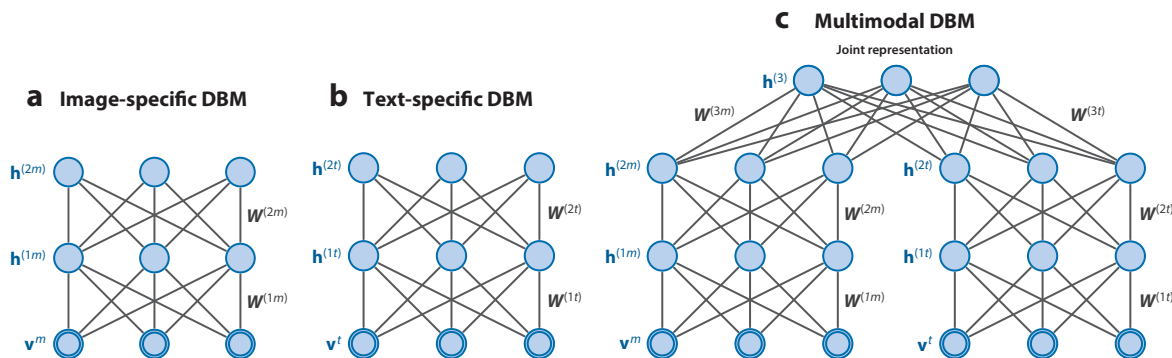
Figure 12

(*a*) Image-specific two-layer deep Boltzmann machine (DBM) that uses a Gaussian model to model the distribution over real-valued image features. (*b*) Text-specific two-layer DBM that uses a replicated softmax model to model its distribution over the word count vectors. (*c*) A multimodal DBM that models the joint distribution over image and text inputs. All but the first (bottom) layer use standard binary variables.

all hidden variables, is written as follows:

$$P(\mathbf{v}^m, \mathbf{v}^t; \theta) = \sum_{\mathbf{h}^{(2m)}, \mathbf{h}^{(2t)}, \mathbf{h}^{(3)}} P(\mathbf{h}^{(2m)}, \mathbf{h}^{(2t)}, \mathbf{h}^{(3)}) \left(\sum_{\mathbf{h}^{(1m)}} P(\mathbf{v}_m, \mathbf{h}^{(1m)} | \mathbf{h}^{(2m)}) \right) \left(\sum_{\mathbf{h}^{(1t)}} P(\mathbf{v}^t, \mathbf{h}^{(1t)} | \mathbf{h}^{(2t)}) \right)$$

$$= \frac{1}{\mathcal{Z}(\theta, M)} \sum_{\mathbf{h}} \exp \Bigg(\underbrace{ - \sum_i \frac{(v_i^m)^2}{2\sigma_i^2} + \sum_{ij} \frac{v_i^m}{\sigma_i} W_{ij}^{(1m)} h_j^{(1m)} + \sum_{jl} W_{jl}^{(2m)} h_j^{(1m)} h_l^{(2m)} }_{\text{Gaussian image pathway}}$$

$$\underbrace{ \sum_{kj} W_{kj}^{(1t)} v_k^t h_j^{(1t)} + \sum_{jl} W_{jl}^{(2t)} h_j^{(1t)} h_l^{(2t)} }_{\text{Replicated softmax text pathway}} + \underbrace{ \sum_{lp} W^{(3t)} h_l^{(2t)} h_p^{(3)} + \sum_{lp} W^{(3m)} h_l^{(2m)} h_p^{(3)} + \sum_p b_p^{(3)} h_p^{(3)} }_{\text{Joint third layer}} \Bigg).$$

$$(42)$$

The normalizing constant depends on the number of words M in the corresponding document because the low-level part of the text pathway contains as many softmax variables as there are words in the document. Similar to the replicated softmax model presented in Section 2.3, the multimodal DBM can be viewed as a family of different-sized DBMs that are created for documents of different lengths that share parameters. Approximate learning and inference can proceed in the same way as discussed in Section 4.1.

As an example, we now consider experimental results of using the MIR-Flickr data set (Huiskes & Lew 2008), which contains one million images retrieved from the social photography website Flickr, along with the user-assigned tags for these images. Bimodal data of this kind have become common in many real-world applications in which we have some image and a few words describing it. There is a need to build representations that fuse this information into a joint space such that each data point can be represented as a single vector. Such representation would be useful for classification and retrieval problems.

Of 1 million images in the MIR-Flickr data set, 25,000 have been annotated using 38 classes including object categories such as "bird," "tree," and "people" and scene categories such as "indoor," "sky," and "night." The remaining 975,000 images were unannotated. We used 10,000 of the 25,000 annotated images for training, 5,000 for validation, and 10,000 for testing, following the experimental setup of Huiskes & Lew (2008).

In our multimodal DBM, the image pathway contained 3,857 linear visible variables[6] and 1,024 $\mathbf{h}^{(1)}$ and 1,024 $\mathbf{h}^{(2)}$ hidden variables. The text pathway consisted of a replicated softmax model with 2,000 visible variables and 1,024 hidden variables, followed by another layer of 1,024 hidden variables. The joint layer contained 2,048 hidden variables, and all hidden variables were binary.

Many real-world applications often have one or more modalities missing. The multimodal DBM can be used to generate such missing data modalities by clamping the observed modalities at the inputs and sampling the hidden modalities by running a standard Gibbs sampler. For example, consider the generation of text conditioned on a given image[7] $\mathbf{v}^m$. The observed modality $\mathbf{v}^m$ is clamped at the inputs, and all hidden variables are initialized randomly. Alternating Gibbs sampling can be used to draw samples (or words) from $P(\mathbf{v}^t | \mathbf{v}^m)$ by updating each hidden layer given the states of the adjacent layers.[8] This process is illustrated for a test image in **Figure 13**,

---

[6]Images were represented by 3,857-dimensional features, which were extracted by concatenating Pyramid Histogram of Words (PHOW) features, Gist, and MPEG-7 descriptors (for details, see Srivastava & Salakhutdinov 2014).

[7]Generation of image features conditioned on text can be done in a similar way.

[8]Remember that the conditional distribution $P(\mathbf{v}^t | \mathbf{h}^{(1t)})$ defines a multinomial distribution over the text vocabulary (see Equation 19). This distribution can then be used to sample words.

	Step 50	Step 100	Step 150	Step 200	Step 250
	travel	beach	sea	water	italy
	trip	ocean	beach	canada	water
	vacation	waves	island	bc	sea
	africa	sea	vacation	britishcolumbia	boat
	earthasia	sand	travel	reflection	italia
	asia	nikon	ocean	alberta	mare
	men	surf	caribbean	lake	venizia
	2007	rocks	tropical	quebec	acqua
	india	coast	resort	ontario	ocean
	tourism	shore	trip	ice	venice

Figure 13

Text generated by the deep Boltzmann machine conditioned on an image by running a Gibbs sampler. The ten words with the highest probability are shown at the end of every 50 sampling steps.

which shows the text generated after every 50 Gibbs steps. Observe that the sampler generates meaningful text while showing some evidence of jumping across different modes. For example, it generates "tropical," "caribbean," and "resort" together, then moves on to "canada," "bc," "quebec lake," "ice," and then to "italia," "venizia," and "mare." Each of these groups of words represent plausible descriptions of the image. Moreover, each group is consistent within itself, suggesting that that the model has been able to associate clusters of consistent descriptions with the same image. The model can also be used to generate images conditioned on text. **Figure 14** shows examples of two such runs.

The same DBM also achieves state-of-the-art classification results on the multimodal MIR-Flickr data set (Huiskes & Lew 2008), compared with linear discriminant analysis (LDA), RBF-kernel support vector machines (SVMs) (Huiskes & Lew 2008), and the multiple kernel learning approach of Guillaumin et al. (2010) (for details, see Srivastava & Salakhutdinov 2014).

6. CONCLUSIONS

This article has reviewed several deep generative models, including DBNs and DBMs. We showed that learning deep generative models that contain many layers of latent variables and millions of parameters can be carried out efficiently. Learned high-level feature representations can be

Input tags	Step 50	Step 100	Step 150	Step 200	Step 250
purple, flowers					
car, automobile					

Figure 14

Images retrieved by running a Gibbs sampler conditioned on the input tags "purple," "flowers" (*top row*) and "car," "automobile" (*bottom row*). The images shown are those which are closest to the sampled image features. Samples were taken after every 50 steps.

successfully applied in a wide spectrum of application domains, including visual object recognition, information retrieval, classification tasks, and regression tasks. Many of the ideas developed in this article are based on the following three crucial principles behind learning deep generative models: First, multiple layers of representation can be greedily learned one layer at a time. Second, greedy learning can be carried out in a completely unsupervised way. Third, a separate fine-tuning stage can be used to further improve either generative or discriminative performance of the final model. Furthermore, using stochastic gradient descent, scaling up learning to billions of data points would not be particularly difficult.

We also developed a new learning algorithm for DBMs. Similar to DBNs, DBMs contain many layers of latent variables. High-level representations are built from large amounts of unlabeled sensory input, and the limited labeled data can then be used to slightly adjust the model parameters for a specific task at hand. We discussed a novel combination of variational and MCMC algorithms for training these Boltzmann machines. When applied to DBMs 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 this algorithm using both the MNIST data set of handwritten digits and the NORB data set of stereo images of 3D objects with highly variable viewpoints and lighting.

Finally, we discussed a DBM model for learning multimodal data representations. Large amounts of unlabeled data can be effectively utilized by the model, in which pathways for each different modality are pretrained independently and later combined together for performing joint learning. The model fuses multiple data modalities into a unified representation, capturing features that are useful for classification and retrieval. Indeed, this model was able to discover a diverse set of plausible descriptions given a test image and to achieve state-of-the-art classification results on the bimodal MIR-Flickr data set.

DISCLOSURE STATEMENT

The author is not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

ACKNOWLEDGMENTS

LITERATURE CITED

Bengio Y. 2009. Learning deep architectures for AI. *Found. Trends Mach. Learn.* 2:1–127

Bengio Y, Lamblin P, Popovici D, Larochelle H. 2007. Greedy layer-wise training of deep networks. *Adv. Neural Inf. Process. Syst.* 19:153–60

Bengio Y, LeCun Y. 2007. Scaling learning algorithms towards AI. In *Large-Scale Kernel Machines*, ed. L Bottou, O Chapelle, D DeCoste, J Weston, pp. 321–360. Cambridge, MA: MIT Press

Blei DM, Ng AY, Jordan MI. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.* 3:993–1022

Blei DM. 2014. Build, compute, critique, repeat: data analysis with latent variable models. *Annu. Rev. Stat. Appl.* 1:203–32

Collobert R, Weston J. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. *Proc. 25th Int. Conf. Mach. Learn. Helsinki*, Jul. 5–9, pp. 160–67. New York: ACM

Dahl GE, Jaitly N, Salakhutdinov R. 2014. Multi-task neural networks for QSAR predictions. arXiv:1406.1231 [stat.ML]

Decoste D, Schölkopf B. 2002. Training invariant support vector machines. *Mach. Learn.* 46:161–90

Guillaumin M, Verbeek J, Schmid C. 2010. Multimodal semi-supervised learning for image classification. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, *San Francisco*, Jun. 13–18, pp. 902–9. Piscataway, NJ: IEEE

Hinton GE. 2002. Training products of experts by minimizing contrastive divergence. *Neural Comput.* 14(8):1711–800

Hinton GE. 2007. To recognize shapes, first learn to generate images. *Prog. Brain Res.* 165:535–47

Hinton GE, Salakhutdinov RR. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–7

Hinton GE, Sejnowski T. 1983. Optimal perceptual inference. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, *Washington, DC*, pp. 448–53. Silver Spring, MD: IEEE

Hinton GE, Osindero S, Teh Y-W. 2006. A fast learning algorithm for deep belief nets. *Neural Comput.* 18(7):1527–54

Hinton G, Deng L, Yu D, Dahl GE, Mohamed A, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.* 29(6):82–97

Hofmann T. 1999. Probabilistic latent semantic analysis. *Proc. 15th Conf. Uncertainty in Artif. Intell.*, *Stockholm, Swe.*, Jul. 30–Aug. 1, pp. 289–96. San Francisco: Morgan Kaufmann

Huiskes MJ, Lew MS. 2008. The MIR Flickr retrieval evaluation. *Proc. 16th Int. Conf. Multimed. Inf. Retr.*, *Vancouver, Can.*, Oct. 30–31. New York: ACM

Krizhevsky A, Sutskever I, Hinton GE. 2012. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* 25:1106–14

Larochelle H, Bengio Y, Louradour J, Lamblin P. 2009. Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.* 10:1–40

LeCun Y, Huang FJ, Bottou L. 2004. Learning methods for generic object recognition with invariance to pose and lighting. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, *Washington, DC*, Jun. 27–Jul. 2, vol. 2, pp. 97–104. Los Alamitos, CA: IEEE

Lee H, Grosse R, Ranganath R, Ng AY. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proc. 26th Intl. Conf. Mach. Learn. Montreal*, Jun. 14–18, pp. 609–16. New York: ACM

Lee TS, Mumford D, Romero R, Lamme V. 1998. The role of the primary visual cortex in higher level vision. *Vision Res.* 38:2429–54

Lenz I, Lee H, Saxena A. 2013. Deep learning for detecting robotic grasps. *Proc. Robot. Sci. Syst. IX Berlin, Ger.*, Jun. 24–28. **http://www.roboticsproceedings.org/rss09/p12.pdf**

Memisevic R, Hinton GE. 2010. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Comput.* 22(6):1473–92

Mohamed A, Dahl GE, Hinton G. 2012. Acoustic modeling using deep belief networks. *IEEE Trans. Audio Speech Lang. Proc.* 20:14–22

Nair V, Hinton GE. 2009. Implicit mixtures of restricted Boltzmann machines. *Adv. Neural Inf. Process. Syst.* 21:1145–52

Neal RM. 2001. Annealed importance sampling. *Stat. Comput.* 11:125–39

Ranzato MA, Huang F, Boureau Y, LeCun Y. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, *Minneapolis, MN*, June 18–23, pp. 1–8. Piscataway, NJ: IEEE. doi: 10.1109/CVPR.2007.383157

Ranzato M, Boureau Y-L, LeCun Y. 2008. Sparse feature learning for deep belief networks. *Adv. Neural Inform. Proc. Syst.* 20:1185–92

Robbins H, Monro S. 1951. A stochastic approximation method. *Ann. Math. Stat.* 22:400–7

Rumelhart DE, Hinton GE, Williams RJ. 1986. Learning representations by back-propagating errors. *Nature* 323:533–36

Salakhutdinov RR. 2008. *Learning and evaluating Boltzmann machines*. Tech. Rep. UTML TR 2008-002, Dep. Comput. Sci., Univ. Toronto, Toronto

Salakhutdinov RR, Hinton GE. 2007. Learning a nonlinear embedding by preserving class neighbourhood structure. *Proc. 11th Int. Conf. Artif. Intell. Stat.*, *San Juan, PR*, Mar. 21–24, pp. 412–19. Brookline, MA: Microtome

Salakhutdinov RR, Hinton GE. 2008. Using deep belief nets to learn covariance kernels for Gaussian processes. *Adv. Neural Inf. Process. Syst.* 20:1249–56

Salakhutdinov RR, Hinton GE. 2009a. Deep Boltzmann machines. *Proc. 12th Int. Conf. Artif. Intell. Stat.*, *Clearwater Beach, FL*, Apr. 16–18, pp. 448–55. Brookline, MA: Microtome

Salakhutdinov RR, Hinton GE. 2009b. Replicated softmax: an undirected topic model. *Adv. Neural Inf. Proc. Syst.* 22:1607–14

Salakhutdinov RR, Hinton GE. 2009c. Semantic hashing. *Int. J. Approx. Reason.* 50:969–78

Salakhutdinov RR, Hinton GE. 2013. Modeling documents with Deep Boltzmann Machines. *Proc. 29th Conf. Uncertain. Artif. Intell.*, *Bellevue, WA*, Jul. 11–15, pp. 616–24. Corvallis, OR: AUAI Press

Salakhutdinov RR, Murray I. 2008. On the quantitative analysis of deep belief networks. *Proc. 25th Int. Conf. Mach. Learn.*, *Helsinki*, Jul. 5–9, pp. 872–79. New York: ACM

Serre T, Oliva A, Poggio TA. 2007. A feedforward architecture accounts for rapid categorization. *PNAS* 104:6424–29

Smolensky P. 1986. Information processing in dynamical systems: foundations of harmony theory. In *Parallel Distributed Processing*, ed. DE Rumelhart, JL McClelland, chapter 6, pp. 194–281. Cambridge, MA: MIT Press

Socher R, Huang EH, Pennington J, Ng AY, Manning CD. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Adv. Neural Inf. Process. Syst.* 24:801–9

Srivastava N, Salakhutdinov R. 2014. Multimodal learning with deep Boltzmann machines. *J. Machine Learn. Res.* 15:2949–80

Sutskever I, Martens J, Dahl G, Hinton G. 2013. On the importance of momentum and initialization in deep learning. *Proc. 30th Int. Conf. Mach. Learn.*, *Atlanta*, Jun. 16–21, pp. 1139–47

Taylor G, Hinton GE, Roweis ST. 2006. Modeling human motion using binary latent variables. *Adv. Neural Inf. Process. Syst.* 19:1345–52

Taylor GW, Fergus R, LeCun Y, Bregler C. 2010. Convolutional learning of spatio-temporal features. *Proc. 11th Eur. Conf. Comput. Vis.*, *Crete, Greece*, Sep. 5–11, pp. 140–153. Berlin: Springer

Tieleman T. 2008. Training restricted Boltzmann machines using approximations to the likelihood gradient. *Proc. 25th Int. Conf. Mach. Learn.*, *Helsinki*, Jul. 5–9, pp. 1064–71. New York: ACM

Torralba A, Fergus R, Weiss Y. 2008. Small codes and large image databases for recognition. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Anchorage, AK*, Jun. 23–28, pp. 1–8. Silver Spring, MD: IEEE. doi: 10.1109/CVPR.2008.4587633

Uria B, Murray I, Larochelle H. 2014. A deep and tractable density estimator. *Proc. 31st Int. Conf. Mach. Learn.*, *Beijing*, Jun. 21–26, pp. 467–75. Brookline, MA: Microtome

Vincent P, Larochelle H, Bengio Y, Manzagol P. 2008. Extracting and composing robust features with denoising autoencoders. *Proc. 25th Int. Conf. Mach. Learn.*, *Helsinki*, Jul. 5–9, pp. 1096–103. New York: ACM

Wang T, Wu D, Coates A, Ng AY. 2012. End-to-end text recognition with convolutional neural networks. *Proc. 21st Int. Conf. Pattern Recognit.*, *Tsukuba, Jpn.*, Nov. 11–15, pp. 3304–8. Piscataway, NJ: IEEE

Welling M, Rosen-Zvi M, Hinton GE. 2005. Exponential family harmoniums with an application to information retrieval. *Adv. Neural Inf. Process. Syst.* 17:1481–88

Younes L. 1989. Parameter inference for imperfectly observed Gibbsian fields. *Probab. Theory Rel. Fields* 82:625–45

Younes L. 1999. On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stoch. Stoch. Rep.* 65:177–228

Yuille AL. 2004. The convergence of Contrastive Divergences. *Adv. Neural Inf. Process. Syst.* 17:1593–600

![Annual Reviews logo](AR logo)

**Annual Review of
Statistics and Its
Application**

Volume 2, 2015

Contents