



An Overview of Restricted Boltzmann Machines

Vidyadhar Upadhyar and P. S. Sastry* 

Abstract | The restricted Boltzmann machine (RBM) is a two-layered network of stochastic units with undirected connections between pairs of units in the two layers. The two layers of nodes are called visible and hidden nodes. In an RBM, there are no connections from visible to visible or hidden to hidden nodes. RBMs are used mainly as a generative model. They can be suitably modified to perform classification tasks also. They are among the basic building blocks of other deep learning models such as deep Boltzmann machine and deep belief networks. The aim of this article is to give a tutorial introduction to the restricted Boltzmann machines and to review the evolution of this model.

1 Introduction

In 1982, Hopfield introduced a fully connected network of interacting units using which one can store and retrieve binary patterns²¹. This network can be regarded as a dynamical system in which the stable states (associated with minima of a suitably defined energy) of the system correspond to the desired memories. The network is initialized with a random state and each node is allowed to change/update its state through a simple rule that depends on the states of nodes connected to it. This dynamics results in a path in the state space that continuously decreases the energy of the network. The Hopfield network was very influential in the development of neural network models in the 1980s. Though successful in storing/retrieving the desired patterns, it was observed that the Hopfield model has limited capacity to store memories and it has the problem of spurious minima. To mitigate some of these problems, a stochastic version of the Hopfield model, called Boltzmann machine, was proposed¹⁷. A Boltzmann machine (BM) is a model of pairwise interacting units where each unit updates its state over time in a probabilistic way depending on the states of the neighboring units. Unlike a Hopfield model, the BM can have some hidden units too. However, learning the parameters of the BM model is computationally intensive. To reduce the complexity of learning, the

connectivity structure of the BM is restricted^{15, 19, 42}. This model is called the restricted Boltzmann machine (RBM). The RBM model has played an important role in the current resurgence of neural networks. The algorithm¹⁸ for unsupervised pretraining of deep belief networks using RBMs is a very significant development in the field in the sense that it rekindled interest in deep neural networks.

RBM is a probabilistic energy-based model (EBM) with a two-layer architecture in which the visible stochastic units are connected to the hidden stochastic units as shown in Fig. 1. There are no connections from visible to visible or hidden to hidden nodes. In a Boltzmann machine, like in a Hopfield model, every unit can be connected to every other unit. RBM is a special case of the Boltzmann machine where an additional bipartite structure is imposed by avoiding the intra-layer connections^{15, 19, 42}. Even though RBM is a generative model, it can also be used as a discriminative model with suitable modifications.

As a generative model, an RBM represents a probability distribution (over states of the visible units) where low-energy configurations have higher probability. The energy is determined by the connection weights which are the parameters to be learnt from the data to build a generative model for the data. The task of learning an RBM requires design of training algorithms and

This article belongs to the Special issue—Recent Advances in Machine Learning.

*Department of Electrical Engineering, Indian Institute of Science, Bangalore, India.
sastry@iisc.ac.in

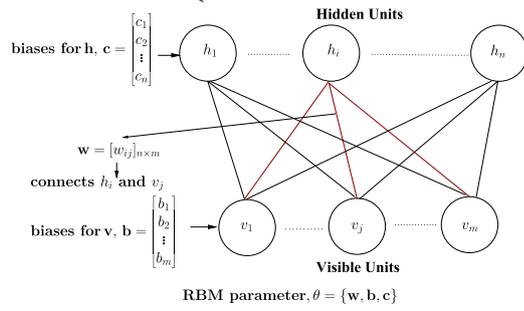


Figure 1: RBM with m visible units and n hidden units. w_{ij} is the weight between h_i and v_j and the terms \mathbf{b} and \mathbf{c} denote the bias for visible and hidden unit, respectively.

a popular method is maximum likelihood estimation of the parameters. RBMs are universal approximators in the sense that any probability distribution over $\{0, 1\}^m$ can be well approximated by an RBM with m visible units and sufficient number of hidden units^{15,24}.

The Boltzmann machine can have all visible nodes. Consider the case with m visible nodes and let $\mathbf{v} \in \{0, 1\}^m$ denotes the state of the visible units. Then, the BM represents a probability distribution given by

$$p(\mathbf{v}|\theta) = \frac{e^{-E(\mathbf{v};\theta)}}{Z}, \tag{1}$$

where $E(\cdot; \theta)$ is the energy function (with parameters θ) and $Z = \sum_{\mathbf{v}} e^{-E(\mathbf{v};\theta)}$ is called the partition function. To increase the expressive power of the model, latent or hidden units can be introduced. Let $\mathbf{h} \in \{0, 1\}^n$ denotes the state of the hidden nodes. The energy function would depend on both visible and hidden units. The probability the model assigns to a given (\mathbf{v}, \mathbf{h}) is

$$p(\mathbf{v}, \mathbf{h}|\theta) = \frac{e^{-E(\mathbf{v}, \mathbf{h};\theta)}}{Z}. \tag{2}$$

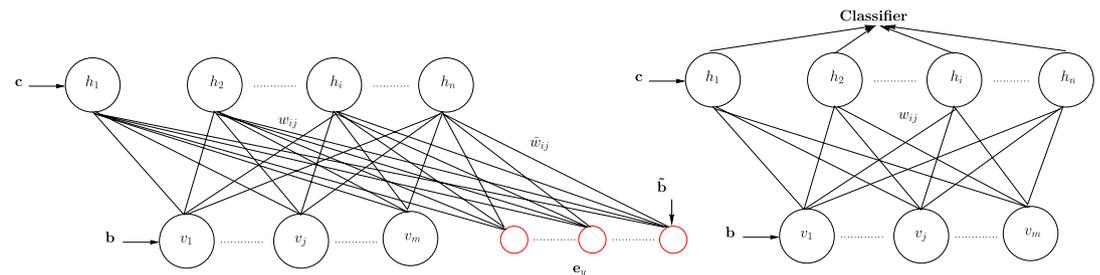


Figure 2: RBM with m visible units and n hidden units. The units representing class label are indicated by the red circles. The class label is represented by these label units through one out of K representations (and we denote it by vector \mathbf{e}_y). w_{ij} is the weight between h_i and v_j , and the terms \mathbf{b} , \mathbf{c} and $\tilde{\mathbf{b}}$ denote the bias for visible, hidden and label units, respectively. \tilde{w}_{ij} is the weight between h_i and j th label unit.

The RBM also has the same distributional form as above. However, there are no connections within the visible and hidden nodes. This would determine the form of the energy function (see Sect. 1.1).

It is also possible to use the RBM as a discriminative model. For this, we learn the joint distribution $p(\mathbf{v}, y)$, where \mathbf{v} is the feature vector and y is the label. The labels are provided along with the training sample as an input to the visible layer as shown in Fig. 2. Once the model is learnt, the new test sample is clamped to the visible units and the conditional distribution of the class given the test sample is used for classifying the test sample. It is also possible to consider the hidden unit representation as features and then building a classifier using them.

The parameters, θ , of the RBM include all the connection weights and the biases. The probability distribution represented by the model is determined by these parameters. Given training data, we need to learn the parameters so that the distribution represented by the model closely matches the desired distribution as indicated by the training data.

The maximum likelihood estimation is the most popular method used to learn RBM parameters. However, the gradient (w.r.t. the parameters of the model) of the log-likelihood is intractable, since it contains an expectation term w.r.t. the model distribution. This expectation is computationally expensive: exponential in (minimum of) the number of visible/hidden units in the model. Therefore, this expectation is approximated by taking an average over the samples from the model distribution. The samples are obtained by exploiting the bipartite connectivity structure of the RBM.

To avoid evaluating intractable expectation in the maximum likelihood method, other methods

such as the pseudo-likelihood, ratio matching and generalized score matching are proposed. These methods use the conditional distributions due to which the intractability referred to above is resolved. However, empirical analysis in Marlin et al.²⁸ showed that the maximum likelihood method is better compared to other methods even though it is computationally intensive.

We would first discuss the model in brief before discussing the learning algorithms.

1.1 The RBM model

The probability assigned to a given visible sample \mathbf{v} by the RBM is

$$p(\mathbf{v}|\theta) = \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h}; \theta)}, \quad (3)$$

where $Z = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h}; \theta)}$ is called the partition function. The energy function is defined depending on the way the visible and hidden units are modeled. For example, the visible and hidden units can be binary or Gaussian. In this article, we mostly consider the case where all units are binary. If both the hidden and visible units are binary, we have $\mathbf{v} = \{0, 1\}^m$ and $\mathbf{h} = \{0, 1\}^n$ and then the energy function is defined as:

$$E(\mathbf{v}, \mathbf{h}; \theta) = - \sum_{i,j} w_{ij} h_i v_j - \sum_{j=1}^m b_j v_j - \sum_{i=1}^n c_i h_i \quad (4)$$

$$= - \mathbf{h}^T \mathbf{w} \mathbf{x} - \mathbf{b}^T \mathbf{x} - \mathbf{c}^T \mathbf{h}. \quad (5)$$

Here, parameter θ consists of $\{\mathbf{w}, \mathbf{b}, \mathbf{c}\}$.

The bipartite structure implies conditional independence of visible units conditioned on all hidden units (and vice-versa). That is,

$$p(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^n p(h_i|\mathbf{v}), \quad p(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^m p(v_i|\mathbf{h}). \quad (6)$$

The above conditional distribution can be expressed as:

$$p(h_i = 1|\mathbf{v}) = \frac{p(h_i = 1, \mathbf{v})}{p(h_i = 0, \mathbf{v}) + p(h_i = 1, \mathbf{v})} = \sigma \left(\sum_{j=1}^m w_{ij} v_j + c_i \right),$$

where $\sigma(x)$ is the logistic sigmoid function, i.e., $\sigma(x) = 1/(1 + e^{-x})$. Similarly, the conditional distribution of the visible unit given the hidden units is:

$$p(v_j = 1|\mathbf{h}) = \sigma \left(\sum_{i=1}^n w_{ij} h_i + b_j \right). \quad (7)$$

The model distribution given in Eq. (3) can be factorized as a mixture of product distributions as:

$$p(\mathbf{v}|\theta) = \frac{1}{Z} \prod_{j=1}^m e^{b_j v_j} \prod_{i=1}^n \left(1 + e^{c_i + \sum_j w_{ij} v_j} \right). \quad (8)$$

The aim is to learn θ such that the model distribution, $p(\mathbf{v}|\theta)$, is close to the data distribution, say, p_{data} in some sense. We discuss the learning algorithms in Sect. 2.

1.2 Representational Power

The representational power of a model defines its ability to capture a class of distributions. As seen from Eq. (8), RBM distribution is a product of mixtures, where each hidden unit contributes to a mixture which is a product of two distributions. The increase in the number of hidden units guarantees improvement in the training log-likelihood or equivalently guarantees reduction in the KL divergence between the data and the model distribution²⁴. Further, they showed that any distribution over $\{0, 1\}^n$ can be approximated arbitrarily well (in terms of the KL divergence measure) with an RBM with $k + 1$ hidden units, where k is the number of input vectors with non-zero probability. This result is generalized to show that any distribution can be approximated arbitrarily well by the RBM with $2^n - 1$ hidden units³². Further, the results are refined³³ to show that RBM with that RBM with $\alpha 2^n - 1$ (where $\alpha < 1$) hidden units are sufficient to approximate any distribution.

2 Learning RBM with Maximum Likelihood

Minimizing the KL divergence (a popular distribution distance measure) between the model and the data distribution is equivalent to maximizing the likelihood of the training samples. Suppose we have N training samples, $\mathcal{T} = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}\}$, the KL divergence between the model and the data distribution is given as:

$$\begin{aligned} d_{\text{KL}}(p_{\text{data}}(\mathbf{v}) \parallel p(\mathbf{v}|\theta)) &= \sum_{\mathbf{v} \in \mathcal{T}} p_{\text{data}}(\mathbf{v}) \log \left(\frac{p_{\text{data}}(\mathbf{v})}{p(\mathbf{v}|\theta)} \right) \\ &= \sum_{\mathbf{v} \in \mathcal{T}} [p_{\text{data}}(\mathbf{v}) \log p_{\text{data}}(\mathbf{v}) - p_{\text{data}}(\mathbf{v}) \log p(\mathbf{v}|\theta)]. \end{aligned} \quad (9)$$

Hence, minimizing KL-divergence is equivalent to maximizing the loglikelihood:

$$\begin{aligned} \arg \min_{\theta} d_{\text{KL}}(p_{\text{data}}(\mathbf{v}) \parallel p(\mathbf{v}|\theta)) \\ = \arg \min_{\theta} \sum_{\mathbf{v} \in \mathcal{T}} [p_{\text{data}}(\mathbf{v}) \log p_{\text{data}}(\mathbf{v}) - p_{\text{data}}(\mathbf{v}) \log p(\mathbf{v}|\theta)] \\ = \arg \max_{\theta} \sum_{\mathbf{v} \in \mathcal{T}} p_{\text{data}}(\mathbf{v}) \log p(\mathbf{v}|\theta), \end{aligned} \tag{10}$$

where $p_{\text{data}}(\mathbf{v}) = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{v} - \mathbf{v}_i)$ is the empirical distribution of the training data and it is not a function of θ . The log-likelihood of θ for a given data vector, \mathbf{v} , can be found by marginalizing out the hidden units from the joint distribution, i.e.,

$$\begin{aligned} \ln \mathcal{L}(\theta|\mathbf{v}) = \ln p(\mathbf{v}|\theta) = \ln \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \\ = \ln \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} - \ln \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}. \end{aligned} \tag{11}$$

The gradient of log-likelihood can be evaluated as:

$$\begin{aligned} \frac{\partial \ln \mathcal{L}(\theta|\mathbf{v})}{\partial \theta} \\ = \frac{\partial}{\partial \theta} \left(\ln \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right) - \frac{\partial}{\partial \theta} \left(\ln \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right) \\ = - \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} + \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \end{aligned} \tag{12}$$

$$\begin{aligned} = - \mathbb{E}_{p(\mathbf{h}|\mathbf{v};\theta)} \left[\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right] + \mathbb{E}_{p(\mathbf{v},\mathbf{h};\theta)} \left[\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right] \\ \triangleq (g(\theta, \mathbf{v}) - f(\theta)), \end{aligned} \tag{13}$$

where \mathbb{E}_p denotes expectation with respect to the distribution p . The first expectation is termed as the positive phase and the second term as the negative phase. The above equation can be simplified using the conditional independent structure of the visible and hidden units. For a specific w_{ij} (similar arguments follow for biases \mathbf{b}, \mathbf{c} also),

$$\begin{aligned} - \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}} \\ = \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) h_i v_j \\ = \sum_{h_i} p(h_i|\mathbf{v}) h_i v_j \sum_{\mathbf{h}_{-i}} p(\mathbf{h}_{-i}|\mathbf{v}) \\ = p(h_i = 1|\mathbf{v}) v_j = \sigma \left(\sum_{j=1}^m w_{ij} v_j + c_i \right) v_j. \end{aligned} \tag{14}$$

Thus, the first term ($g(\theta, \mathbf{v})$) can be easily obtained. However, the second term is exponential in the size of the smallest layer ($2^{\min(m,n)}$) which follows from the equations below.

$$\begin{aligned} \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} &= \sum_{\mathbf{v}} p(\mathbf{v}) \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \\ &= \sum_{\mathbf{h}} p(\mathbf{h}) \sum_{\mathbf{v}} p(\mathbf{v}|\mathbf{h}) \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}. \end{aligned} \tag{15}$$

For a given set of training samples, $\mathcal{T} = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}\}$, the log-likelihood gradient is

$$\begin{aligned} \frac{1}{N} \sum_{l=1}^N \frac{\partial \ln \mathcal{L}(\theta|\mathbf{v}^{(l)})}{\partial \theta} &= \frac{1}{N} \sum_{l=1}^N \\ &\times \left[-\mathbb{E}_{p(\mathbf{h}|\mathbf{v}^{(l)})} \left[\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right] + \mathbb{E}_{p(\mathbf{v},\mathbf{h};\theta)} \left[\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right] \right]. \end{aligned}$$

Since the empirical data distribution, p_{data} , assigns probability of $1/N$ to each sample, the summation of the first term above is expectation with respect to the data distribution. Specifically, the updates for weights and biases are

$$\begin{aligned} \Delta w_{ij} &= \frac{\partial}{\partial w_{ij}} (-\ln p(\mathbf{v}|\theta)) \\ &= \mathbb{E}_{p_{\text{model}}} [v_j h_i] - \mathbb{E}_{p_{\text{data}}} [v_j h_i] \\ \Delta b_j &= \frac{\partial}{\partial b_j} (-\ln p(\mathbf{v}|\theta)) \\ &= \mathbb{E}_{p_{\text{model}}} [v_j] - \mathbb{E}_{p_{\text{data}}} [v_j] \\ \Delta c_i &= \frac{\partial}{\partial c_i} (-\ln p(\mathbf{v}|\theta)) \\ &= \mathbb{E}_{p_{\text{model}}} [h_i] - \mathbb{E}_{p_{\text{data}}} [h_i]. \end{aligned}$$

However, expectation under the model distribution, $p(\mathbf{v}, \mathbf{h}; \theta)$, is computationally intractable since the number of terms in the expectation summation grows exponentially with the (minimum of) the number of hidden units/visible units present in the model. Hence, Markov chain Monte Carlo sampling methods are used to obtain expectation under the model distribution.

2.1 Markov Chain Monte Carlo Estimation and Gibbs Sampling

The estimation of expectation of a function, $g : S \rightarrow R$, w.r.t. a given probability distribution p (defined on S) can be obtained through independent samples drawn from the distribution p as:

$$\mu = \mathbb{E}_p[g(x)] \approx \frac{1}{N} \sum_{i=1}^N g(x_i) \tag{16}$$

where $x_i \in S$ and $x_i \sim p(\cdot)$.

The estimate above is called the Monte Carlo estimate. Suppose obtaining samples from the distribution $p(\cdot)$ is difficult or the distribution is known only upto a normalizing constant. Then, it is advantageous to design a Markov chain with $p(\cdot)$ as the stationary distribution and obtain samples from the stationary distribution of that Markov chain to approximate the expectation as in Eq. (16). This is called the Markov chain Monte Carlo (MCMC) estimation. Unlike the samples obtained in the Monte Carlo estimation case, the MCMC samples are, in general, not independent.

It still remains to specify how to construct a Markov chain which converges to the required distribution. The Gibbs sampler considers univariate conditional distributions where all of the random variables but one are assigned fixed values. Such univariate distributions are easier to simulate than complex joint distributions. The sampling is done by simulating n random variables sequentially from the n univariate conditionals. More specifically, the Gibbs sampler generates the next state \mathbf{x}^{s+1} from the current state of the chain \mathbf{x}^s as follows.

1. $x_1^{s+1} \sim p(x_1|x_2^s, x_3^s, \dots, x_n^s)$,
2. $x_2^{s+1} \sim p(x_2|x_1^{s+1}, x_3^s, x_4^s, \dots, x_n^s)$,
3. $x_i^{s+1} \sim p(x_i|x_{1:i-1}^{s+1}, x_{i+1:n}^s)$,
for $i = 3, \dots, n - 1$,
4. $x_n^{s+1} \sim p(x_n|x_1^{s+1}, x_2^{s+1}, \dots, x_{n-1}^{s+1})$,

where $p(x_i|\cdot)$ are conditionals obtained from the target distribution $p(\cdot)$.

For the RBM case, all of the hidden units can be sampled simultaneously, because they are conditionally independent from each other given the visible units. Similarly, all of the visible units can be sampled simultaneously, since they are conditionally independent from each other given the hidden units. This sampling method which update many variables simultaneously is called block Gibbs sampling. As seen earlier, the conditionals are given by sigmoid functions.

2.2 Contrastive Divergence

The MCMC methods work well if the samples are obtained when the Markov chain converges to the stationary distribution. The samples are needed for every iteration of the gradient ascent on the

log-likelihood. The computational cost becomes large if one waits for the chain to converge at each iteration. To reduce the computational cost, Markov chain has to be initialized close to the model distribution. One way to achieve this is by initializing the Markov chain with the samples from the data distributon. This algorithm is called contrastive divergence algorithm¹⁹, which is a popular algorithm to learn RBM. In this algorithm, a single sample, obtained after running a Markov chain, initialized with data samples, for K steps, is used to approximate the expectation as:

$$\begin{aligned} \nabla_{\theta} f(\theta) &= -\mathbb{E}_{p(\mathbf{v}, \mathbf{h}; \theta)} [\nabla_{\theta} E(\mathbf{v}, \mathbf{h}; \theta)] \\ &= -\mathbb{E}_{p(\mathbf{v}; \theta)} \mathbb{E}_{p(\mathbf{h}|\mathbf{v}; \theta)} [\nabla_{\theta} E(\mathbf{v}, \mathbf{h}; \theta)] \\ &\approx -\mathbb{E}_{p(\mathbf{h}|\tilde{\mathbf{v}}^{(K)}; \theta)} [\nabla_{\theta} E(\tilde{\mathbf{v}}^{(K)}, \mathbf{h}; \theta)] \tag{17} \\ &\triangleq \hat{f}'(\theta, \tilde{\mathbf{v}}^{(K)}). \end{aligned}$$

Here, $\tilde{\mathbf{v}}^{(K)}$ is the sample obtained after K transitions of the Markov chain (defined by the current parameter values θ) initialized with the training sample \mathbf{v} . In practice, the mini-batch version of this algorithm is used. The algorithm is called CD- k if we run the Markov chain for k steps.

It can be interpreted as minimizing the difference between the two Kullback–Leibler divergences¹⁹, i.e.,

$$\begin{aligned} d_{\text{KL}}(p_{\text{data}}(\mathbf{v}) \parallel p(\mathbf{v}|\theta)) &= d_{\text{KL}}(p_{\text{data}}(\mathbf{v}) \parallel p(\mathbf{v}|\theta)) \\ &\quad - d_{\text{KL}}(p_k(\mathbf{v}) \parallel p(\mathbf{v}|\theta)), \end{aligned} \tag{18}$$

where p_k is the distribution of the chain after k steps. As the parameter $k \rightarrow \infty$, the CD- k algorithm is equivalent to ML. However, in practice, a small k is used. As the learning progresses, the mixing rate of the Markov chain decreases rapidly⁵⁰. This algorithm is effective in many scenarios in approximating the direction of gradient ascent. However, it is shown in many empirical studies that the likelihood can diverge on specific training sets after an initial increase in likelihood^{10, 13, 41}.

The fixed points of CD differ from those of ML and CD- k gives a biased estimate of the gradient. This bias depends on the number of units in the RBM and the maximum change in energy that can be produced by changing a single unit¹⁴. The bias is also affected by the distance in variation between the model distribution and the initial distribution of the Gibbs chain¹⁴. The studies in^{1, 5} reveal that the bias in CD- k approximation can lead to convergence to parameters that do not reach the maximum likelihood. The CD- k update is not a gradient of any function, and counter-intuitive regularization function that causes CD

learning to cycle indefinitely is constructed in⁴³. The CD-*k* approximation can also be viewed as a stochastic approximation algorithm. The convergence conditions are analyzed in^{22,50,51}.

2.2.1 Other Approaches

There are two main classes of approaches to make the learning of RBM more efficient. The first is to design an efficient MCMC method to get good representative samples from the model distribution and thereby reduce the variance of the estimated gradient. The persistent contrastive divergence (PCD) algorithm⁴⁵ was initially proposed to train Boltzmann machines⁴⁹, then later shown to perform better than CD in the context of RBM⁴⁵. The algorithm is similar to CD learning. The only change is in the Gibbs chain initialization. Specifically, the Gibbs chain is not reinitialized to the training vector after *k*-steps for each parameter update; instead, it starts the chain at samples from the previous iteration (termed fantasy particles). In the fast persistent contrastive divergence (FPCD) algorithm⁴⁶, an additional set of weights are introduced and shown to perform better by improving the mixing rate of the persistent chain. The regular and the fast weights both contribute to the effective weight update. The fast weights are also calculated similar to the regular updates, but with much stronger weight decay and a faster learning rate. Other such algorithms based on modifying the MCMC sampling procedure include the population (pop-CD)³⁶, and average contrastive divergence (ACD)²⁶. Another popular algorithm, parallel tempering (PT)⁹, is also based on MCMC. However, in general, such advanced MCMC methods are computationally intensive.

The second approach is to design better optimization strategies which are robust to the noise in estimated gradient^{4,11,29}. Most approaches to design better optimization methods for learning RBMs are second-order optimization techniques that either need approximate Hessian inverse or an estimate of the inverse Fisher matrix. The AdaGrad¹² method uses diagonal approximation of the Hessian matrix, while TONGA³⁸ assumes block diagonal structure. The Hessian-Free (H-F) method²⁹ is an iterative procedure which approximately solves a linear system to obtain the curvature through matrix-vector product. The H-F method is used to design natural gradient descent for learning Boltzmann machines¹¹. A sparse Gaussian graphical model can be used to estimate the inverse fisher matrix to devise factorized natural gradient descent procedure¹⁶. All

these methods either need additional computations to solve an auxiliary linear system or are computationally intensive methods to directly estimate the inverse Fisher matrix. The centered gradients (CG) method³⁰ is motivated by the principle that by removing the mean of the training data and the mean of the hidden activations from the visible and the hidden variables, respectively, the conditioning of the underlying optimizing problem can be improved³¹. The RBM log-likelihood function is a difference of convex functions, since both *f* and *g* in Eq. (13) can be written as log-sum-exponential function form. This property is exploited in⁴ to devise a majorization-minimization optimization algorithm called the stochastic spectral descent (SSD) algorithm. A stochastic variation of difference of convex programming (DCP) can also be used to exploit the difference of convex functions property of the RBM log-likelihood function^{35,47}.

There are methods which avoid evaluation of the partition function by modifying the cost function; for example, pseudolikelihood, ratio matching and score matching methods. Suppose we have *N* training samples, $\mathcal{T} = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}\}$, then the learning proceeds by maximizing/minimizing the following objective functions. (In the equations below \mathbf{v}_{-d} denotes all visible units excluding the *d*th unit.)

- Pseudolikelihood:

$$L_{\theta}^{PL}(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}) = \frac{1}{Nm} \sum_{i=1}^N \sum_{j=1}^m \log p(v_j^{(i)} | \mathbf{v}_{-j}^{(i)}, \theta) \tag{19}$$

- Ratio matching:

$$L_{\theta}^{RM}(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}) = -\frac{1}{Nm} \sum_{i=1}^N \sum_{j=1}^m \left(1 - p(v_j^{(i)} | \mathbf{v}_{-j}^{(i)}, \theta)\right)^2 \tag{20}$$

- Generalized score matching

$$L_{\theta}^{GSM}(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}) = -\frac{1}{Nm} \sum_{i=1}^N \sum_{j=1}^m \left(\frac{1}{p(v_j^{(i)} | \mathbf{v}_{-j}^{(i)}, \theta)} - \frac{1}{p_{\text{data}}(v_j^{(i)} | \mathbf{v}_{-j}^{(i)}, \theta)} \right)^2 \tag{21}$$

The main motivation for all these approaches is to avoid the intractability associated with the

gradient of the log partition function. These methods avoid the estimation of the gradient of the log partition function, since they use the conditional distribution, i.e., $p(v_j|\mathbf{v}_{-j})$. In the conditional distribution, the partition function appears both in the numerator and denominator and hence cancels out. While these approaches result in computationally efficient learning algorithms, their performance is not always comparable to that of CD- k .

3 Generalization

So far, we have considered RBM model where both visible and hidden nodes are binary. When visible nodes are binary, an RBM can model a distribution only over $\{0, 1\}^m$. When we need a generative model for continuous data, we need continuous valued visible units. The basic RBM model can be generalized to take care of such situations. For example, the visible units can be modified to have Gaussian distribution. If we consider Gaussian visible units and binary hidden units, i.e., $\mathbf{v} \in \mathcal{R}^{N_v}$ and $\mathbf{h} \in \{0, 1\}^{N_h}$ then the energy function can be defined as⁶,

$$E(\mathbf{v}, \mathbf{h}|\theta) = \sum_i \frac{(v_i - b_i)^2}{2\sigma^2} - \sum_{i,j} \frac{w_{ij}v_i h_j}{\sigma^2} - \sum_j c_j h_j,$$

where σ^2 is the variance associated with the Gaussian visible unit. This model is the most popular Gaussian RBM (GRBM). The maximum likelihood estimation-based learning of GRBM is more involved than that of binary RBM. There are many attempts to make the learning more efficient by defining the energy function associated with the GRBM in several different ways^{23, 48}. For natural images, sparse penalty was shown to provide meaningful representations for natural images²⁵. However, simple mixture models outperform the GRBM models in terms of final likelihood values⁴⁴. With the reparameterization of energy function and an improved learning algorithm the GRBM model is shown to extract meaningful representations⁶. The analysis in Wang et al.⁴⁸ showed that the failures in learning GRBMs are due to the inefficiency in training algorithms rather than the model itself. Several training recipes based on the knowledge of the data distribution were also suggested. These algorithms require carefully chosen learning rate, since a large learning rate results in divergence of the log-likelihood and a small learning rate leads to very slow convergence. One needs

careful parameter initialization and restricting of the gradient during the update, for avoiding the divergence of the log-likelihood. The GRBM model is not efficient in modeling natural images, as the extracted hidden features do not represent sharp edges occurring at the object boundaries. Also, these features are not expressive enough to gain some advantages in classification tasks³⁷. Based on this, some modifications of the GRBM are suggested whereby the hidden units can play a role in modeling of covariances of the visible units³⁷. It is argued in Courville et al.⁷ that the difficiency in GRBM is because of binary hidden units. That paper proposed spike and slab RBM in which hidden units are modeled as the element-wise product of a real-valued vector with a binary vector. Each hidden unit is associated with a binary spike variable and the real vector-valued slab variable. There are many such generalizations of RBMs proposed.

Another interesting generalization is to the case where the visible units are not binary but take only finitely many values. We mention one such model. A family of RBMs with shared parameters called the Replicated Softmax model²⁰, extracts semantic representations from an unstructured collection of documents. Each RBM has softmax visible units that takes on values in some discrete alphabet, i.e., $\mathbf{v} \in \{1, 2, \dots, K\}^N$. The hidden units are binary and we have $\mathbf{h} \in \{0, 1\}^F$. We can essentially use the binary RBM by coding the visible unit states as ‘one-of- K ’ binary vectors. Let \mathbf{V} be a $N \times K$ observed binary matrix with $v_{ik} = 1$ if visible unit i takes on the k th value. The energy function is defined as,

$$E(\mathbf{V}, \mathbf{h}; \theta) = - \sum_{i=1}^N \sum_{j=1}^F \sum_{k=1}^K W_{ijk} h_j v_{ik} - \sum_{i=1}^N \sum_{k=1}^K v_{ik} b_{ik} - N \sum_{j=1}^F h_j c_j \tag{22}$$

Thus, essentially the binary RBM method can take care of such cases.

4 Multi-layered Networks with RBM

One of the developments that contributed to the initial success of deep learning is greedy layer-wise training of stacked RBMs which provided good initialization for training neural networks with many hidden layers. The RBMs can be stacked to form multiple layers yielding different models. For example, the deep belief network and the deep Boltzmann machine have RBM as a building block but their connections differ.

4.1 Deep Belief Network

Deep belief network (DBN) is a generative model containing many layers of hidden units, in which each layer captures correlations between the activities of hidden units in the layer below. The top two layers form an undirected bipartite graph similar to RBM. The lower layers form a directed sigmoid belief network, as shown in Fig. 3. A DBN with only one hidden layer is an RBM.

The joint distribution represented by the DBN shown in Fig. 3 consisting of two hidden layers and a visible layer, is given by,

$$p(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \theta) = p(\mathbf{v}|\mathbf{h}^1; \theta) p(\mathbf{h}^1, \mathbf{h}^2; \theta), \tag{23}$$

where $\theta = \{\mathbf{w}^1, \mathbf{w}^2, \mathbf{b}, \mathbf{c}_1, \mathbf{c}_2\}$. A deep, hierarchical model can be learnt through layer-by-layer training where each pair of consecutive layers can be considered as an RBM. More specifically, the bottom layer is first trained to maximize $\mathbb{E}_{p_{\text{data}}}[\log p(\mathbf{v})]$ using CD or PCD. Once the parameters of the bottom layer are learnt, the hidden unit samples are generated by clamping the visible unit with the data. These samples serve as the data for training the next layer. This can be thought of as approximately maximizing $\mathbb{E}_{p_{\text{data}}(\mathbf{v})} \mathbb{E}_{p^{(1)}(h^{(1)}|\mathbf{v})}[\log p^{(2)}(h^{(1)})]$, where $p^{(1)}$ and $p^{(2)}$ are the distributions represented by the first and second RBM, respectively. This learning algorithm can be repeated for as many layers as required. The above training procedure increases the variational lower bound on the log-likelihood of the data as the number of layers increases¹⁸.

The learnt weights of the DBN are considered as initial weights of the MLP which consists of an additional classification layer. This MLP is fine-tuned for the classification task. The initial success of DBN is due to this algorithm which provided an efficient way to initialize the weights of an MLP and thereby improve the performance of MLP on many discriminative tasks both in terms of training time and the classification accuracy¹⁸.

4.2 Deep Boltzmann Machine

The deep Boltzmann machine (DBM) is also a generative model which has RBM as a building block. As mentioned earlier, learning fully connected Boltzmann machine is computationally intensive. Therefore, the hidden units are stacked in a layer-wise manner where each layer learns internal representations that become increasingly complex capturing higher-order correlations. Unlike DBN, in DBM all the connections are undirected as shown in Fig. 4.

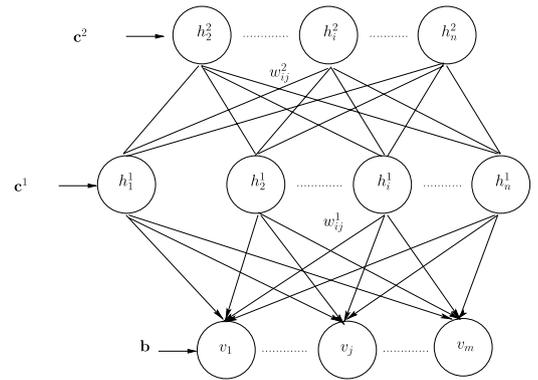


Figure 3: Deep belief net with two hidden layers denoted as $\{\mathbf{h}^1, \mathbf{h}^2\}$.

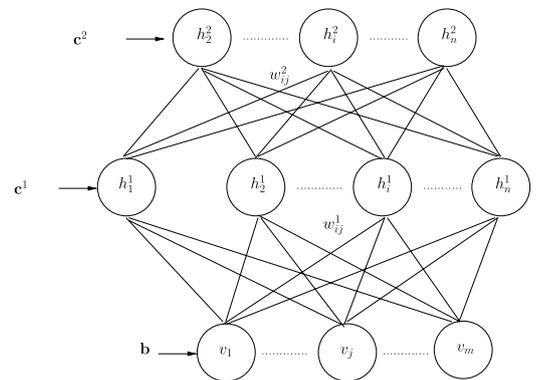


Figure 4: Deep Boltzmann machine with two hidden layers denoted as $\{\mathbf{h}^1, \mathbf{h}^2\}$.

The energy function of the DBM shown in Fig. 4 is,

$$E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \theta) = -\mathbf{v}^T \mathbf{W}^1 \mathbf{h}^1 - \mathbf{h}^1^T \mathbf{W}^2 \mathbf{h}^2, \tag{24}$$

where $\theta = \{\mathbf{W}^1, \mathbf{W}^2\}$ are the weights connecting visible to hidden and hidden to hidden layers. The learning of DBM is similar to the learning of DBN given in 4.1

5 Evaluation

Since RBM is a generative model, the learnt RBM is to be evaluated based on the distribution learnt. Normally, the quality of the learnt RBM is evaluated based on the average log-likelihood on a test sample. The average log-likelihood on N test samples is given by,

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \log p(\mathbf{v}^{(i)}), \tag{25}$$

where $p(\mathbf{v}^{(i)})$ is the probability (or the likelihood) of the i th test sample, $\mathbf{v}^{(i)}$. The model with higher average test log-likelihood is better. The test log-likelihood can also be used in devising a stopping criterion for the learning and for fixing the hyper-parameters through cross validation.

The likelihood $p(\mathbf{v})$ can be written as $p^*(\mathbf{v})/Z$. While $p^*(\mathbf{v})$ is easy to evaluate, the normalizing constant Z , called the partition function, is computationally expensive. Therefore, various sampling-based estimators have been proposed for the estimation of the log-likelihood. The first approach is to approximately estimate the partition function using the samples obtained from the model distribution. Since sampling from the model distribution of an RBM is complicated, a useful sampling technique is the importance sampling where samples obtained from a simple distribution, called the proposal distribution, are used to estimate the partition function.

5.1 Simple Importance Sampling

The expectation of a function $f(x)$ with respect to a given distribution $p(x)$ can be approximated by taking average over independent samples from $p(x)$. When the distribution is too complex (rugged and high dimensional), generating independent samples is difficult. In such situations, a simple distribution from which independent samples can easily be generated is used to assist in obtaining the approximate expectation. The algorithm works as follows.

Suppose it is possible to generate independent samples from a distribution $q(x)$, which can be written as $q(x) = \frac{q^*(x)}{Z_q}$. Here, Z_q is the normalizing constant given as $Z_q = \int q^*(x) dx$. We assume that we know $p(x)$ upto a normalizing constant, i.e., $p(x) = \frac{p^*(x)}{Z_p}$ and it is easy to calculate $p^*(x)$. Now we can write expectation of $f(x)$ with respect to $p(x)$ as,

$$\mathbb{E}_p[f(x)] = \int f(x) \frac{p^*(x)}{Z_p} dx \tag{26}$$

To eliminate Z_p in the above equation consider the following ratio,

$$\begin{aligned} \frac{Z_p}{Z_q} &= \int \frac{p^*(x)}{Z_q} dx \\ &= \int \frac{p^*(x)}{q^*(x)} q(x) dx \end{aligned} \tag{27}$$

Now, substituting the value of Z_p in Eq. (26) yields,

$$\begin{aligned} \mathbb{E}_p[f(x)] &= \int f(x) \frac{p^*(x)}{Z_q \int \frac{p^*(y)}{q^*(y)} q(y) dy} dx \\ &= \int f(x) \frac{p^*(x) q(x)}{\int \frac{p^*(y)}{q^*(y)} q(y) dy} dx \end{aligned} \tag{28}$$

The above equation can be written in terms of the N samples obtained from the distribution $q(x)$ as,

$$\begin{aligned} \mathbb{E}_p[f(x)] &\approx \frac{\sum_{i=1}^N w_i f(x_i)}{\sum_{i=1}^N w_i}, \\ \text{where } x_i &\sim q(x) \text{ and } w_i = \frac{p^*(x_i)}{q^*(x_i)} \end{aligned} \tag{29}$$

Note that this method can also be used to estimate the partition function as given in Eq. (27),

$$\hat{z} = \frac{Z_p}{Z_q} = \frac{1}{N} \sum_{i=1}^N w_i \tag{30}$$

It is known that for a high-dimensional setting, the variance of \hat{z} is very high and can be infinite sometimes when the proposal distribution, $q(x)$, is not a good approximation of the true distribution²⁷.

5.2 Annealed Importance Sampling

The importance sampling method-based partition function estimator suffers from a large variance if the proposal distribution is not a good approximation to the target distribution. This is due to the large variance of importance weights. To overcome the issue with choosing the proposal distribution in importance sampling algorithm, a sequence of intermediate probability distributions can be used to move gradually³⁴. If the two consecutive intermediate distributions differ by a small amount, then the variance of importance weights will be under control. The intermediate distributions p_0, p_1, \dots, p_k , with $p_0 = p_A(\mathbf{x})$ (proposal distribution) and $p_k = p_B(\mathbf{x})$ (target distribution), are chosen such that they should satisfy the following properties.

- $p_k(\mathbf{x}) \neq 0$ whenever $p_{k+1}(\mathbf{x}) \neq 0$.
- The unnormalized probability $p_k^*(\mathbf{x})$ is easy to calculate $\forall k$.
- For each k , it is possible to get sample \mathbf{x}' given \mathbf{x} through Markov transition $T_k(\mathbf{x}'|\mathbf{x})$ which leaves $p_k(\mathbf{x})$ invariant.

Then, the successive application of importance sampling method through the sequence of distributions allows one to find expectations with respect to the target distribution. Annealed importance sampling is often used for estimating the average test log-likelihood of RBM models.

There are other approaches to estimate the average test log-likelihood directly by marginalizing over the hidden variables from the model distribution. However, the computational complexity grows exponentially with the minimum of the number of visible and hidden units present in the model. Hence, an approximate method which uses a sample-based estimator called conservative sampling-based likelihood estimator (CSL)². A more efficient method called reverse annealed importance sampling estimator (RAISE)³ implements CSL by formulating the problem of marginalization as a partition function estimation problem.

6 Conclusions

The RBM model has played an important role in the recent spectacular developments in neural network models and deep learning. The RBM is a generative model and it can extract patterns from the given data in an unsupervised manner. It is a very useful method for unsupervised feature learning. RBMs have also played an important role in unsupervised pretraining for weight initialization of deep neural networks. As mentioned earlier, the RBM is a building block for DBNs and DBMs. The pre-trained networks are used as an efficient initialization of MLPs which are fine-tuned later for the specific tasks. RBMs have been used in a variety of applications such as collaborative filtering³⁹, to analyze connectivity structure of brain using fMRI images⁴⁰, constructing topic models from unstructured text data²⁰, modeling of natural images⁸, etc.

In this paper, we presented a tutorial introduction to the RBM model and provided some discussion on learning an RBM from data through maximum likelihood estimation. We described the popular CD-*k* algorithm and also discussed many other methods that were proposed. We also discussed generalizations of the basic model both in terms of real-valued visible units as well as multilayer networks constructed using RBMs.

Many other successful variants of RBM have also been proposed to further improve the ability of the model in representing different types of data. For instance, Conditional restricted Boltzmann machine (CRBM) was proposed for collaborative filtering, Gaussian–Bernoulli restricted

Boltzmann machine (GRBM) was proposed for real-valued continuous data, recurrent temporal Boltzmann machine (RTBM) was devised to represent sequential data, and convolutional RBMs are developed to capture the spatial structure in images and to learn acoustic filters from spectrogram of speech data.

RBMs represent useful generative models and unlike some of the other generative models (e.g., GANs), RBMs are also very effective as discriminative models. However, learning an RBM is computationally expensive. This is mainly because of the intractability of the partition function. As discussed here, one uses the MCMC sampling techniques to estimate gradients of the likelihood function for learning. Better methods for learning RBMs is currently an important research problem. The problem of constructing RBMs with real-valued visible units is also a problem of much current interest. Designing proper architectures for RBMs with multiple hidden layers is also an interesting problem. RBMs, along with CNNs, provided the initial push for the deep learning revolution over the past few years and are likely to be important for the field in the years to come.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 26 December 2018 Accepted: 30 January 2019
Published online: 18 February 2019

References

- 1 Bengio Y, Delalleau O (2009) Justifying and generalizing contrastive divergence. *Neural Comput* 21(6):1601–1621
- 2 Bengio Y, Yao L, Cho K (2013) Bounding the test log-likelihood of generative models. [arXiv:1311.6184](https://arxiv.org/abs/1311.6184) (arXiv preprint)
- 3 Burda Y, Grosse RB, Salakhutdinov R (2014) Accurate and conservative estimates of MRF log-likelihood using reverse annealing. [arXiv:1412.8566](https://arxiv.org/abs/1412.8566) (arXiv preprint)
- 4 Carlson D, Cevher V, Carin L (2015) Stochastic spectral descent for restricted Boltzmann machines. In: *Proceedings of the eighteenth international conference on artificial intelligence and statistics*, pp 111–119
- 5 Carreira-PMA, Hinton GE (2005) On contrastive divergence learning. In: *Proceedings of the tenth international workshop on artificial intelligence and statistics*. Citeseer, pp 33–40
- 6 Cho K, Ilin A, Raiko T (2011) Improved learning of Gaussian–Bernoulli restricted Boltzmann machines. In: Honkela T, Duch W, Girolami M, Kaski S (eds) *Artificial neural networks and machine learning—ICANN 2011*. Springer, Berlin, pp 10–17 (ISBN 978-3-642-21735-7)

- 7 Courville A, Bergstra J, Bengio Y A spike and slab restricted Boltzmann machine. In: Gordon G, Dunson D, Dudik M (eds) Proceedings of the fourteenth international conference on artificial intelligence and statistics, volume 15 of proceedings of machine learning research, Fort Lauderdale, FL, USA, 11–13 Apr 2011a. PMLR, pp 233–241. <http://proceedings.mlr.press/v15/courville1a.html>
- 8 Courville Aaron, Bergstra James, Bengio Yoshua (2011b) Unsupervised models of images by spike-and-slab rbms. In: Proceedings of the 28th international conference on machine learning, ICML'11, USA. Omnipress, pp 1145–1152. <http://dl.acm.org/citation.cfm?id=3104482.3104626> (ISBN 978-1-4503-0619-5)
- 9 Desjardins G, Courville A, Bengio Y (2010a) Adaptive parallel tempering for stochastic maximum likelihood learning of RBMS. [arXiv:1012.3476](https://arxiv.org/abs/1012.3476) (arXiv preprint)
- 10 Desjardins G, Courville AC, Bengio Y, Vincent P, Delalleau O (2010b) Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines. In: International conference on artificial intelligence and statistics, pp 145–152
- 11 Desjardins G, Pascanu R, Courville AC, Bengio Y (2013) Metric-free natural gradient for joint-training of Boltzmann machines. *CoRR*. [arXiv:1301.3545](https://arxiv.org/abs/1301.3545)
- 12 Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. *J Mach Learn Res* 12(Jul):2121–2159
- 13 Fischer A, Igel C (2010) Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines. In: Artificial neural networks–ICANN 2010. Springer, pp 208–217
- 14 Fischer A, Igel C (2011) Bounding the bias of contrastive divergence learning. *Neural Comput* 23(3):664–673
- 15 Freund Y, Haussler D (1994) Unsupervised learning of distributions of binary vectors using two layer networks. Computer Research Laboratory [University of California, Santa Cruz]
- 16 Grosse RB, Salakhutdinov R (2015) Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In: Proceedings of the 32nd international conference on machine learning, volume 37, ICML'15, pp 2304–2313. *JMLR.org*. <http://dl.acm.org/citation.cfm?id=3045118.3045363>
- 17 Hinton GE, Sejnowski TJ (1986) Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter learning and relearning in Boltzmann machines. MIT Press, Cambridge, pp 282–317. URL <http://dl.acm.org/citation.cfm?id=104279.104291> (ISBN 0-262-68053-X)
- 18 Hinton G, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. *Neural Comput* 18(7):1527–1554
- 19 Hinton GE (2002) Training products of experts by minimizing contrastive divergence. *Neural Comput* 14(8):1771–1800
- 20 Hinton GE, Salakhutdinov RR (2009) Replicated Softmax: an undirected topic model. In: Bengio Y, Schuurmans D, Lafferty JD, Williams CKI, Culotta A (eds) Advances in neural information processing systems 22. Curran Associates, Inc., pp 1607–1614. <http://papers.nips.cc/paper/3856-replicated-softmax-an-undirected-topic-model.pdf>
- 21 Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci* 79(8):2554–2558. <https://doi.org/10.1073/pnas.79.8.2554>. <https://www.pnas.org/content/79/8/2554> (ISSN 0027-8424)
- 22 Jiang B, Wu T-Y, Jin Y, Wong WH (2016) Convergence of contrastive divergence algorithm in exponential family. [arXiv:1603.05729](https://arxiv.org/abs/1603.05729) (arXiv e-prints)
- 23 Krizhevsky A (2009) Learning multiple layers of features from tiny images. Master's Thesis. <http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- 24 Le Roux N, Bengio Y (2008) Representational power of restricted Boltzmann machines and deep belief networks. *Neural Comput* 20(6):1631–1649
- 25 Lee H, Ekanadham C, Ng AY (2008) Sparse deep belief net model for visual area v2. In: Platt JC, Koller D, Singer Y, Roweis ST (eds) Advances in neural information processing systems 20. Curran Associates, Inc, pp 873–880. <http://papers.nips.cc/paper/3313-sparse-deep-belief-net-model-for-visual-area-v2.pdf>
- 26 Ma X, Wang X (2016) Average contrastive divergence for training restricted Boltzmann machines. *Entropy* 18(1):35
- 27 MacKay DJC (2003) Information theory, inference, and learning algorithms, vol 7. Cambridge University Press, Cambridge
- 28 Marlin BM, Swersky K, Chen B, Freitas ND (2010) Inductive principles for restricted Boltzmann machine learning. In: International conference on artificial intelligence and statistics, pp 509–516
- 29 Martens J (2010) Deep learning via hessian-free optimization. In: ICML
- 30 Melchior J, Fischer A, Wiskott L (2016) How to center deep Boltzmann machines. *J Mach Learn Res* 17(99):1–61
- 31 Montavon G, Klaus-Robert M (2012) Deep Boltzmann machines and the centering trick. Springer, Berlin, pp 621–637. https://doi.org/10.1007/978-3-642-35289-8_33 (ISBN 978-3-642-35289-8)
- 32 Montufar G, Ay N (2011) Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines. *Neural Comput* 23(5):1306–1319. https://doi.org/10.1162/neco_a_00113. https://doi.org/10.1162/NECO_a_00113 (ISSN 0899-7667)
- 33 Montufar G, Rauh J (2017) Hierarchical models as marginals of hierarchical models. *Int J Approx Reason* 88:531–546. <https://doi.org/10.1016/j.ijar.2016.09.003>. <http://www.sciencedirect.com/science/article/pii/S0888613X16301414> (ISSN 0888-613X)
- 34 Neal RM (2001) Annealed importance sampling. *Stat Comput* 11(2):125–139

- 35 Nitanda A, Suzuki T Stochastic difference of convex algorithm and its application to training deep Boltzmann machines. In: Singh A, Zhu J (eds) Proceedings of the 20th international conference on artificial intelligence and statistics, vol 54 of Proceedings of machine learning research, Fort Lauderdale, FL, USA, 20–22 Apr 2017, pp 470–478. PMLR. <http://proceedings.mlr.press/v54/nitanda17a.html>
- 36 Oswin K, Igel C, Fischer A (2015) Population-contrastive-divergence: does consistency help with RBM training? CoRR. [arXiv:1510.01624](https://arxiv.org/abs/1510.01624)
- 37 Ranzato M, Hinton GE (2010) Modeling pixel means and covariances using factorized third-order Boltzmann machines. In: 2010 IEEE computer society conference on computer vision and pattern recognition, pp 2551–2558. <https://doi.org/10.1109/CVPR.2010.5539962>
- 38 Roux NL, Manzagol PA, Bengio Y (2008) Topmoumoute online natural gradient algorithm. In: Platt JC, Koller D, Singer Y, Roweis ST (eds) Advances in neural information processing systems 20. Curran Associates, Inc., pp 849–856. <http://papers.nips.cc/paper/3234-topmoumoute-online-natural-gradient-algorithm.pdf>
- 39 Salakhutdinov R, Mnih A, Hinton G (2007) Restricted Boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on machine learning, ICML '07, New York, NY, USA. ACM, pp 791–798. <https://doi.org/10.1145/1273496.1273596>. <http://doi.acm.org/10.1145/1273496.1273596>(ISBN 978-1-59593-793-3)
- 40 Schmäh T, Hinton GE, Small SL, Strother S, Zemel RS (2009) Generative versus discriminative training of RBMs for classification of fMRI images. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in neural information processing systems 21. Curran Associates, Inc., pp 1409–1416. <http://papers.nips.cc/paper/3577-generative-versus-discriminative-training-of-rbms-for-classification-of-fmri-images.pdf>
- 41 Schulz H, Müller A, Behnke S (2010) Investigating convergence of restricted Boltzmann machine learning. In: NIPS 2010 workshop on deep learning and unsupervised feature learning
- 42 Smolensky P (1986) Information processing in dynamical systems: foundations of harmony theory
- 43 Sutskever I, Tieleman T (2010) On the convergence properties of contrastive divergence. In: International conference on artificial intelligence and statistics, pp 789–795
- 44 Theis L, Gerwin S, Sinz F, Bethge M (2011). In: All likelihood, deep belief is not enough. *J Mach Learn Res* 12:3071–3096. <http://dl.acm.org/citation.cfm?id=1953048.2078204>(ISSN 1532-4435)
- 45 Tieleman T (2008) Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th international conference on machine learning. ACM, pp 1064–1071
- 46 Tieleman T, Hinton G (2009) Using fast weights to improve persistent contrastive divergence. In: Proceedings of the 26th annual international conference on machine learning. ACM, pp 1033–1040
- 47 Upadhy V, Sastry PS (2017) Learning RBM with a DC programming approach. In: Proceedings of the ninth Asian conference on machine learning, volume 77 of proceedings of machine learning research. PMLR, 15–17 Nov 2017, pp 498–513
- 48 Wang N, Melchior J, Wiskott L (2014) Gaussian-binary restricted Boltzmann machines on modeling natural image statistics. CoRR. [arXiv:1401.5900](https://arxiv.org/abs/1401.5900)
- 49 Younes L (1989) Parametric inference for imperfectly observed gibbsian fields. *Prob Theory Relat Fields* 82(4):625–645
- 50 Younes L (1999) On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stoch Stoch Rep* 65(3–4):177–228. <https://doi.org/10.1080/17442509908834179>
- 51 Yuille AL (2006) The convergence of contrastive divergences. Department of Statistics, UCLA



Vidyadhar Upadhy received his master's degree from the Electrical Engineering Department, Indian Institute of Technology Madras. He is currently pursuing Ph.D. in the Electrical Engineering Department, Indian Institute of Science, Bangalore. His research interests include pattern recognition and machine learning.



P. S. Sastry received his B.Sc. in physics from Indian Institute of Technology, Kharagpur, and B.E. in electrical communication engineering and Ph.D. from the Department Of Electrical Engineering, both from Indian Institute of Science (IISc),

Bangalore. He is currently a professor in the Department of Electrical Engineering, IISc. He was the chairman of the department during 2010–2015. He has held visiting positions at the University of Massachusetts, Amherst; University of Michigan, Ann Arbor; General Motors Research Laboratories, Warren; and Texas A&M University, College Station. His research interests include pattern recognition, machine learning, data mining and computational neuroscience. Prof. Sastry received the C. V. Raman Award for Young Scientists from the Government of Karnataka, Hari Om Ashram Dr. Vikram Sarabhai Research Award from PRL, Ahmadabad, Most Valued Colleague Award from General Motors Corporation, and the Alumni Award for Excellence in Research from IISc. He is a Fellow of the Indian National Academy of Engineering and the National Academy of Sciences, India.