An Introduction to Restricted Boltzmann Machines

Asja Fischer[1,2] and Christian Igel[2]

[1] Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany
[2] Department of Computer Science, University of Copenhagen, Denmark

Abstract. Restricted Boltzmann machines (RBMs) are probabilistic graphical models that can be interpreted as stochastic neural networks. The increase in computational power and the development of faster learning algorithms have made them applicable to relevant machine learning problems. They attracted much attention recently after being proposed as building blocks of multi-layer learning systems called deep belief networks. This tutorial introduces RBMs as undirected graphical models. The basic concepts of graphical models are introduced first, however, basic knowledge in statistics is presumed. Different learning algorithms for RBMs are discussed. As most of them are based on Markov chain Monte Carlo (MCMC) methods, an introduction to Markov chains and the required MCMC techniques is provided.

1 Introduction

Boltzmann machines (BMs) have been introduced as bidirectionally connected networks of stochastic processing units, which can be interpreted as neural network models [1,16]. A BM can be used to learn important aspects of an unknown probability distribution based on samples from this distribution. In general, this learning process is difficult and time-consuming. However, the learning problem can be simplified by imposing restrictions on the network topology, which leads us to *restricted Boltzmann machines* (RBMs, [34]), the topic of this tutorial.

A (restricted) BM is a parameterized generative model representing a probability distribution. Given some observations, the training data, learning a BM means adjusting the BM parameters such that the probability distribution represented by the BM fits the training data as well as possible. Boltzmann machines consist of two types of units, so called visible and hidden neurons, which can be thought of as being arranged in two layers. The visible units constitute the first layer and correspond to the components of an observation (e.g., one visible unit for each pixel of a digital input image). The hidden units model dependencies between the components of observations (e.g., dependencies between pixels in images). They can be viewed as non-linear feature detectors [16].

Boltzmann machines can also be regarded as particular graphical models [22], more precisely undirected graphical models also known as Markov random fields. The embedding of BMs into the framework of probabilistic graphical models provides immediate access to a wealth of theoretical results and well-developed

L. Alvarez et al. (Eds.): CIARP 2012, LNCS 7441, pp. 14–36, 2012.
© Springer-Verlag Berlin Heidelberg 2012

algorithms. Therefore, our tutorial introduces RBMs from this perspective. Computing the likelihood of an undirected model or its gradient for inference is in general computationally intensive, and this also holds for RBMs. Thus, sampling based methods are employed to approximate the likelihood and its gradient. Sampling from an undirected graphical model is in general not straightforward, but for RBMs Markov chain Monte Carlo (MCMC) methods are easily applicable in the form of Gibbs sampling, which will be introduced in this tutorial along with basic concepts of Markov chain theory.

After successful learning, an RBM provides a closed-form representation of the distribution underlying the observations. It can be used to compare the probabilities of (unseen) observations and to sample from the learnt distribution (e.g., to generate image textures [25,21]), in particular from marginal distributions of interest. For example, we can fix some visible units corresponding to a partial observation and sample the remaining visible units for completing the observation (e.g., to solve an image inpainting task [21]).

Boltzmann machines have been proposed in the 1980s [1,34]. Compared to the times when they were first introduced, RBMs can now be applied to more interesting problems due to the increase in computational power and the development of new learning strategies [15]. Restricted Boltzmann machines have received a lot of attention recently after being proposed as building blocks of multi-layer learning architectures called deep belief networks (DBNs, [19,17]). The idea is that the hidden neurons extract relevant features from the observations. These features can serve as input to another RBM. By stacking RBMs in this way, one can learn features from features in the hope of arriving at a high level representation.

It is an important property that single as well as stacked RBMs can be reinterpreted as deterministic feed-forward neural networks. Than they are used as functions from the domain of the observations to the expectations of the latent variables in the top layer. Such a function maps the observations to learnt features, which can, for example, serve as input to a supervised learning system. Further, the neural network corresponding to a trained RBM or DBN can be augmented by an output layer, where units in the new added output layer represent labels corresponding to observations. Then the model corresponds to a standard neural network for classification or regression that can be further trained by standard supervised learning algorithms [31]. It has been argued that this initialization (or unsupervised pretraining) of the feed-forward neural network weights based on a generative model helps to overcome problems observed when training multi-layer neural networks [19].

This introduction to RBMs is meant to supplement existing tutorials, such as the highly recommended review by Bengio [2], by providing more background information on Markov random fields and MCMC methods in Section 2 and Section 3, respectively. However, basic knowledge in statistics is presumed. We put an emphasis on topics that are – based on our experience – sometimes not familiar to people starting with RBMs. Restricted Boltzmann machines will be presented in Section 4. Section 5 will consider RBM training algorithms based

on approximations of the log-likelihood gradient. This includes a discussion of contrastive divergence learning [15] as well as parallel tempering [10]. We will close by hinting at generalizations of RBMs in sections 6 and 7.

2 Graphical Models

Probabilistic graphical models describe probability distributions by mapping conditional dependence and independence properties between random variables on a graph structure (two sets of random variables $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are conditionally independent given a set of random variables $\boldsymbol{X}_3$ if $p(\boldsymbol{X}_1, \boldsymbol{X}_2|\boldsymbol{X}_3) = p(\boldsymbol{X}_1|\boldsymbol{X}_3)p(\boldsymbol{X}_2|\boldsymbol{X}_3)$). Visualization by graphs can help to develop, understand and motivate probabilistic models. Furthermore, complex computations (e.g., marginalization) can be derived efficiently by using algorithms exploiting the graph structure.

There exist graphical models associated with different kind of graph structures, for example *factor graphs*, *Bayesian networks* associated with directed graphs, and *Markov random fields*, which are also called *Markov networks* or undirected graphical models. This tutorial focuses on the latter. A general introduction to graphical models for machine learning can, for example be found in [5]. The most comprehensive resource on graphical models is the textbook by Koller and Friedman [22].

2.1 Undirected Graphs and Markov Random Fields

First, we will summarize some fundamental concepts from graph theory. An *undirected graph* is a tuple $G = (V, E)$, where V is a finite set of nodes and E is a set of undirected edges. An edge consists out of a pair of nodes from V. If there exists an edge between two nodes v and w, i.e. $\{v, w\} \in E$, w belongs to the neighborhood of v and vice versa. The *neighborhood* $\mathcal{N}_v := \{w \in V : \{w, v\} \in E\}$ of v is defined by the set of nodes connected to v. A *clique* is a subset of V in which all nodes are pairwise connected. A clique is called *maximal* if no node can be added such that the resulting set is still a clique. In the following we will denote by $\mathcal{C}$ the set of all maximal cliques of an undirected graph. We call a sequence of nodes $v_1, v_2, \ldots, v_m \in V$, with $\{v_i, v_{i+1}\} \in E$ for $i = 1, \ldots, m-1$ a *path* from v_1 to v_m. A set $\mathcal{V} \subset V$ *separates* two nodes $v \notin \mathcal{V}$ and $w \notin \mathcal{V}$, if every path from v to w contains a node from $\mathcal{V}$.

We now associate a random variable X_v taking values in a state space Λ_v with each node v in an undirected graph $G = (V, E)$. To ease the notation, we assume $\Lambda_v = \Lambda$ for all $v \in V$. The random variables $\boldsymbol{X} = (X_v)_{v \in V}$ are called *Markov random field* (MRF) if the joint probability distribution p fulfills the *(global) Markov property* w.r.t. the graph: For all disjunct subsets $\mathcal{A}, \mathcal{B}, \mathcal{S} \subset V$, where all nodes in $\mathcal{A}$ and $\mathcal{B}$ are separated by $\mathcal{S}$ the variables $(X_a)_{a \in \mathcal{A}}$ and $(X_b)_{b \in \mathcal{B}}$ are conditional independent given $(X_s)_{s \in \mathcal{S}}$, i.e. for all $\boldsymbol{x} \in \Lambda^{|V|}$ it holds $p((x_a)_{a \in \mathcal{A}}|(x_t)_{t \in \mathcal{S} \cup \mathcal{B}}) = p((x_a)_{a \in \mathcal{A}}|(x_t)_{t \in \mathcal{S}})$. A set of nodes $\mathrm{MB}(v)$ is called the *Markov blanket* of node v, if for any set of nodes $\mathcal{B}$ with $v \notin \mathcal{B}$ we have

$p(v \mid \mathrm{MB}(v), \mathcal{B}) = p(v \mid \mathrm{MB}(v))$. This means that v is conditional independent from any other variables given $\mathrm{MB}(v)$. In an MRF, the Markov blanket $\mathrm{MB}(v)$ is given by the neighborhood $\mathcal{N}_v$ of v, a fact that is also referred to as *local* Markov property.

Since conditional independence of random variables and the factorization properties of the joint probability distribution are closely related, one can ask if there exists a general factorization form of the distributions of MRFs. An answer to this question is given by the *Hammersley-Clifford Theorem* (for rigorous formulations and proofs we refer to [23,22]). The theorem states that a strictly positive distribution p satisfies the Markov property w.r.t. an undirected graph G if and only if p factorizes over G. A distribution is said to factorize about an undirected graph G with maximal cliques $\mathcal{C}$ if there exists a set of non-negative functions $\{\psi_C\}_{C \subset \mathcal{C}}$, called *potential functions*, with

$$\forall \boldsymbol{x}, \hat{\boldsymbol{x}} \in \Lambda^{|V|} : (x_c)_{c \in C} = (\hat{x}_c)_{c \in C} \Rightarrow \psi_C(\boldsymbol{x}) = \psi_C(\hat{\boldsymbol{x}}) \tag{1}$$

and

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}). \tag{2}$$

The normalization constant $Z = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$ is called *partition function*.

If p is strictly positive, the same holds for the potential functions. Thus we can write

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C) = \frac{1}{Z} e^{\sum_{C \in \mathcal{C}} \ln \psi_C(\boldsymbol{x}_C)} = \frac{1}{Z} e^{-E(\boldsymbol{x})} \quad , \tag{3}$$

where we call $E := \sum_{C \in \mathcal{C}} \ln \psi_C(\boldsymbol{x}_C)$ the *energy function*. Thus, the probability distribution of every MRF can be expressed in the form given by (3), which is also called *Gibbs distribution*.

2.2 Unsupervised Learning

Unsupervised learning means learning (important aspects of) an unknown distribution q based on sample data. This includes finding new representations of data that foster learning, generalization, and communication. If we assume that the structure of the graphical model is known and the energy function belongs to a known family of functions parameterized by $\boldsymbol{\theta}$, unsupervised learning of a data distribution with an MRF means adjusting the parameters $\boldsymbol{\theta}$. We write $p(\boldsymbol{x}|\boldsymbol{\theta})$ when we want to emphasize the dependency of a distribution on its parameters.

We consider training data $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell\}$. The data samples are assumed to be independent and identically distributed (i.i.d.). That is, they are drawn independently from some unknown distribution q. A standard way of estimating the parameters of a statistical model is maximum-likelihood estimation. Applied to MRFs, this corresponds to finding the MRF parameters that maximize the probability of S under the MRF distribution, i.e. training corresponds to finding the parameters $\boldsymbol{\theta}$ that maximize the likelihood given the training data. The

likelihood $\mathcal{L} : \Theta \to \mathbb{R}$ of an MRF given the data set S maps parameters $\boldsymbol{\theta}$ from a parameter space Θ to $\mathcal{L}(\boldsymbol{\theta} \,|\, S) = \prod_{i=1}^{\ell} p(\boldsymbol{x}_i|\boldsymbol{\theta})$. Maximizing the likelihood is the same as maximizing the log-likelihood given by

$$\ln \mathcal{L}(\boldsymbol{\theta} \,|\, S) = \ln \prod_{i=1}^{\ell} p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{i=1}^{\ell} \ln p(\boldsymbol{x}_i|\boldsymbol{\theta}) \ . \tag{4}$$

For the Gibbs distribution of an MRF it is in general not possible to find the maximum likelihood parameters analytically. Thus, numerical approximation methods have to be used, for example gradient ascent which is described below.

Maximizing the likelihood corresponds to minimizing the distance between the unknown distribution q underlying S and the distribution p of the MRF in terms of the *Kullback-Leibler-divergence* (KL-divergence), which for a finite state space Ω is given by:

$$\mathrm{KL}(q||p) = \sum_{\boldsymbol{x} \in \Omega} q(\boldsymbol{x}) \ln \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} = \sum_{\boldsymbol{x} \in \Omega} q(\boldsymbol{x}) \ln q(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in \Omega} q(\boldsymbol{x}) \ln p(\boldsymbol{x}) \tag{5}$$

The KL-divergence is a (non-symmetric) measure of the difference between two distributions. It is always positive and zero if and only if the distributions are the same. As becomes clear by equation (5) the KL-divergence can be expressed as the difference between the entropy of q and a second term. Only the latter depends on the parameters subject to optimization. Approximating the expectation over q in this term by the training samples from q results in the log-likelihood. Therefore, maximizing the log-likelihood corresponds to minimizing the KL-divergence.

Optimization by Gradient Ascent. If it is not possible to find parameters maximizing the likelihood analytically, the usual way to find them is gradient ascent on the log-likelihood. This corresponds to iteratively updating the parameters $\boldsymbol{\theta}^{(t)}$ to $\boldsymbol{\theta}^{(t+1)}$ based on the gradient of the log-likelihood. Let us consider the following update rule:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \underbrace{\eta \frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} \left(\sum_{i=1}^{N} \ln \mathcal{L}(\boldsymbol{\theta}^{(t)}|\boldsymbol{x}_i) \right) - \lambda \boldsymbol{\theta}^{(t)} + \nu \Delta \boldsymbol{\theta}^{(t-1)}}_{:= \Delta \boldsymbol{\theta}^{(t)}} \tag{6}$$

If the constants $\lambda \in \mathbb{R}_0^+$ and $\nu \in \mathbb{R}_0^+$ are set to zero, we have vanilla gradient ascent. The constant $\eta \in \mathbb{R}^+$ is the learning rate. As we will see later, it can be desirable to strive for models with weights having small absolute values. To achieve this, we can optimize an objective function consisting of the log-likelihood minus half of the norm of the parameters $\|\theta\|^2/2$ weighted by λ. This method called *weight decay* penalizes weights with large magnitude. It leads to the $-\lambda \boldsymbol{\theta}^{(t)}$ term in our update rule (6). In a Bayesian framework, weight decay

can be interpreted as assuming a zero-mean Gaussian prior on the parameters. The update rule can be further extended by a *momentum* term $\Delta\boldsymbol{\theta}^{(t-1)}$, weighted by the parameter ν. Using a momentum term helps against oscillations in the iterative update procedure and can speed-up the learning process as known from feed-forward neural network training [31].

Introducing Latent Variables. Suppose we want to model an m-dimensional unknown probability distribution q (e.g., each component of a sample corresponds to one of m pixels of an image). Typically, not all variables $\boldsymbol{X} = (X_v)_{v \in V}$ in an MRF need to correspond to some observed component, and the number of nodes is larger than m. We split $\boldsymbol{X}$ into *visible* (or *observed*) variables $\boldsymbol{V} = (V_1, \ldots, V_m)$ corresponding to the components of the observations and *latent* (or *hidden*) variables $\boldsymbol{H} = (H_1, \ldots, H_n)$ given by the remaining $n = |V| - m$ variables. Using latent variables allows to describe complex distributions over the visible variables by means of simple (conditional) distributions. In this case the Gibbs distribution of an MRF describes the joint probability distribution of $(\boldsymbol{V}, \boldsymbol{H})$ and one is usually interested in the marginal distribution of $\boldsymbol{V}$ which is given by

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \ , \tag{7}$$

where $Z = \sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})}$. While the visible variables correspond to the components of an observation, the latent variables introduce dependencies between the visible variables (e.g., between pixels of an input image).

Log-Likelihood Gradient of MRFs with Latent Variables. Restricted Boltzmann machines are MRFs with hidden variables and RBM learning algorithms are based on gradient ascent on the log-likelihood. For a model of the form (7) with parameters $\boldsymbol{\theta}$, the log-likelihood given a single training example $\boldsymbol{v}$ is

$$\ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v}) = \ln p(\boldsymbol{v} \,|\, \boldsymbol{\theta}) = \ln \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} = \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} - \ln \sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \tag{8}$$

and for the gradient we get:

$$\begin{aligned} \frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \right) - \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \right) \\ &= -\frac{1}{\sum\limits_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \frac{1}{\sum\limits_{\boldsymbol{v}, \boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})}} \sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \\ &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}, \boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \end{aligned} \tag{9}$$

In the last step we used that the conditional probability can be written in the following way:

$$p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \frac{p(\boldsymbol{v}, \boldsymbol{h})}{p(\boldsymbol{v})} = \frac{\frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \tag{10}$$

Note that the last expression of equality (9) is the difference of two expectations: the expected values of the energy function under the model distribution and under the conditional distribution of the hidden variables given the training example. Directly calculating this sums, which run over all values of the respective variables, leads to a computational complexity which is in general exponential in the number of variables of the MRF. To avoid this computational burden, the expectations can be approximated by samples drawn from the corresponding distributions based on MCMC techniques.

3 Markov Chains and Markov Chain Monte Carlo Techniques

Markov chains play an important role in RBM training because they provide a method to draw samples from 'complex' probability distributions like the Gibbs distribution of an MRF. This section will serve as an introduction to some fundamental concepts of Markov chain theory. A detailed introduction can be found, for example, in [6] and the aforementioned textbooks [5,22]. The section will describe Gibbs sampling as an MCMC technique often used for MRF training and in particular for training RBMs.

3.1 Definition of a Markov Chain and Convergence to Stationarity

A *Markov chain* is a time discrete stochastic process for which the *Markov property* holds, that is, a family of random variables $X = \{X^{(k)} | k \in \mathbb{N}_0\}$ which take values in a (in the following considerations finite) set Ω and for which $\forall k \geq 0$ and $\forall j, i, i_0, \ldots, i_{k-1} \in \Omega$ it holds

$$p_{ij}^{(k)} := P\left(X^{(k+1)} = j | X^{(k)} = i, X^{(k-1)} = i_{k-1}, \ldots, X^{(0)} = i_0\right) \tag{11}$$

$$= P\left(X^{(k+1)} = j | X^{(k)} = i\right) \quad . \tag{12}$$

This means that the next state of the system depends only on the current state and not on the sequence of events that preceded it. If for all $k \geq 0$ the $p_{ij}^{(k)}$ have the same value p_{ij}, the chain is called *homogeneous* and the matrix $\mathbf{P} = (p_{ij})_{i,j \in \Omega}$ is called *transition matrix* of the homogeneous Markov chain.

If the starting distribution $\mu^{(0)}$ (i.e., the probability distribution of $X^{(0)}$) is given by the probability vector $\boldsymbol{\mu}^{(0)} = (\mu^{(0)}(i))_{i \in \Omega}$, with $\mu^{(0)}(i) = P(X^{(0)} = i)$, the distribution $\boldsymbol{\mu}^{(k)}$ of $X^{(k)}$ is given by $\boldsymbol{\mu}^{(k)\,\mathrm{T}} = \boldsymbol{\mu}^{(0)\,\mathrm{T}} \mathbf{P}^k$.

A distribution π for which it holds $\boldsymbol{\pi}^{\mathrm{T}} = \boldsymbol{\pi}^{\mathrm{T}} \mathbf{P}$ is called *stationary distribution*. If the Markov chain for any time k reaches the stationary distribution $\boldsymbol{\mu}^{(k)} = \boldsymbol{\pi}$ all subsequent states will be distributed accordingly, that is, $\boldsymbol{\mu}^{(k+n)} = \boldsymbol{\pi}$ for

all $n \in \mathbb{N}$. A sufficient (but not necessary) condition for a distribution π to be stationary w.r.t. a Markov chain described by the transition probabilities $p_{ij}, i, j \in \Omega$ is that $\forall i, j \in \Omega$ it holds:

$$\pi(i)p_{ij} = \pi(j)p_{ji} \ . \tag{13}$$

This is called the *detailed balance condition*.

Especially relevant are Markov chains for which it is known that there exists an unique stationary distribution. For finite Ω this is the case if the Markov chain is *irreducible*. A Markov chain is irreducible if one can get from any state in Ω to any other in a finite number of transitions or more formally $\forall i, j \in \Omega$ $\exists k > 0$ with $P(X^{(k)} = j | X^{(0)} = i) > 0$.

A chain is called *aperiodic* if for all $i \in \Omega$ the greatest common divisor of $\{k | P(X^{(k)} = i | X^{(0)} = i) > 0 \wedge k \in \mathbb{N}_0\}$ is 1. One can show that an irreducible and aperiodic Markov chain on a finite state space is guarantied to converge to its stationary distribution (see, e.g., [6]). That is, for an arbitrary starting distribution μ it holds

$$\lim_{k \to \infty} d_V(\boldsymbol{\mu}^T \, \mathrm{P}^k, \boldsymbol{\pi}^T) = 0 \ , \tag{14}$$

where d_V is the *distance of variation*. For two distributions α and β on a finite state space Ω, the distance of variation is defined as

$$d_V(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}|\boldsymbol{\alpha} - \boldsymbol{\beta}| = \frac{1}{2} \sum_{x \in \Omega} |\alpha(x) - \beta(x)| \ . \tag{15}$$

To ease the notation, we allow both row and column probability vectors as arguments of the functions in (15).

Markov chain Monte Carlo methods make use of this convergence theorem for producing samples from certain probability distribution by setting up a Markov chain that converges to the desired distributions. Suppose you want to sample from a distribution q with a finite state space. Then you construct an irreducible and aperiodic Markov chain with stationary distribution $\pi = q$. This is a non-trivial task. If t is large enough, a sample $X^{(t)}$ from the constructed chain is then approximately a sample from π and therefore from q. Gibbs Sampling [13] is such a MCMC method and will be described in the following section.

3.2 Gibbs Sampling

Gibbs Sampling belongs to the class of Metropolis-Hastings algorithms [14]. It is a simple MCMC algorithm for producing samples from the joint probability distribution of multiple random variables. The basic idea is to update each variable subsequently based on its conditional distribution given the state of the others. We will describe it in detail by explaining how Gibbs sampling can be used to simulate the Gibbs distribution of an MRF.

We consider an MRF $\boldsymbol{X} = (X_1, \ldots, X_N)$ w.r.t. a graph $G = (V, E)$, where $V = \{1, \ldots, N\}$ for the sake of clearness of notation. The random variables X_i, $i \in V$ take values in a finite set Λ and $\pi(\boldsymbol{x}) = \frac{1}{Z} e^{-\mathcal{E}(\boldsymbol{x})}$ is the joint probability

distribution of $\boldsymbol{X}$. Furthermore, if we assume that the MRF changes its state during time, we can consider $X = \{\boldsymbol{X}^{(k)} | k \in \mathbb{N}_0\}$ as a Markov chain taking values in $\Omega = \Lambda^N$ where $\boldsymbol{X}^{(k)} = (X_1^{(k)}, \ldots, X_N^{(k)})$ describes the state of the MRF at time $k \geq 0$. At each transition step we now pick a random variable X_i, $i \in V$ with a probability $q(i)$ given by a strictly positive probability distribution q on V and sample a new value for X_i based on its conditional probability distribution given the state $(x_v)_{v \in V \setminus i}$ of all other variables $(X_v)_{v \in V \setminus i}$, i.e. based on $\pi\left(X_i | (x_v)_{v \in V \setminus i}\right) = \pi\left(X_i | (x_w)_{w \in \mathcal{N}_i}\right)$. Therefore, the transition probability p_{xy} for two states $\boldsymbol{x}, \boldsymbol{y}$ of the MRF $\boldsymbol{X}$ with $\boldsymbol{x} \neq \boldsymbol{y}$ is given by:

$$p_{\boldsymbol{xy}} = \begin{cases} q(i)\pi\left(y_i | (x_v)_{v \in V \setminus i}\right), & \text{if } \exists i \in V \text{ so that } \forall v \in V \text{ with } v \neq i\text{: } x_v = y_v \\ 0, & \text{else .} \end{cases} \tag{16}$$

And the probability, that the state of the MRF $\boldsymbol{x}$ stays the same, is given by:

$$p_{\boldsymbol{xx}} = \sum_{i \in V} q(i)\pi\left(x_i | (x_v)_{v \in V \setminus i}\right) \quad . \tag{17}$$

It is easy to see that the joint distribution π of the MRF is the stationary distribution of the Markov chain defined by these transition probabilities by showing that the detailed balance condition (13) holds: For $\boldsymbol{x} = \boldsymbol{y}$ this follows directly. If $\boldsymbol{x}$ and $\boldsymbol{y}$ differ in the value of more than one random variable it follows from the fact that $p_{\boldsymbol{yx}} = p_{\boldsymbol{xy}} = 0$. Assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ differ only in the state of exactly one variable X_i, i.e., $y_j = x_j$ for $j \neq i$ and $y_i \neq x_i$, then it holds:

$$\begin{aligned} \pi(\boldsymbol{x})p_{\boldsymbol{xy}} = \pi(\boldsymbol{x})q(i)\pi\left(y_i | (x_v)_{v \in V \setminus i}\right) &= \pi\left(x_i, (x_v)_{v \in V \setminus i}\right) q(i) \frac{\pi\left(y_i, (x_v)_{v \in V \setminus i}\right)}{\pi\left((x_v)_{v \in V \setminus i}\right)} \\ &= \pi\left(y_i, (x_v)_{v \in V \setminus i}\right) q(i) \frac{\pi\left(x_i, (x_v)_{v \in V \setminus i}\right)}{\pi\left((x_v)_{v \in V \setminus i}\right)} \\ &= \pi(\boldsymbol{y})q(i)\pi\left(x_i | (x_v)_{v \in V \setminus i}\right) = \pi(\boldsymbol{y})p_{\boldsymbol{yx}}. \end{aligned} \tag{18}$$

Since π is strictly positive so are the conditional probability distributions of the single variables. Thus, it follows that every single variable X_i can take every state $x_i \in \Lambda$ in a single transition step and thus every state of the whole MRF can reach any other in Λ^N in a finite number of steps and the Markov chain is irreducible. Furthermore it follows from the positivity of the conditional distributions that $p_{\boldsymbol{xx}} > 0$ for all $\boldsymbol{x} \in \Lambda^N$ and thus that the Markov chain is aperiodic. Aperiodicity and irreducibility guaranty that the chain converges to the stationary distribution π.

In practice the single random variables to be updated are usually not chosen at random based on a distribution q but subsequently in fixed predefined order. The corresponding algorithm is often referred to as *periodic Gibbs Sampler*. If $\boldsymbol{P}$ is the transition matrix of the Gibbs chain, the convergence rate of the periodic

Gibbs sampler to the stationary distribution of the MRF is bounded by the following inequality (see for example [6]):

$$|\boldsymbol{\mu}\mathbf{P}^k - \boldsymbol{\pi}| \leq \frac{1}{2}|\boldsymbol{\mu} - \boldsymbol{\pi}|(1 - e^{-N\triangle})^k, \tag{19}$$

where $\triangle = \sup_{l \in V} \delta_l$ and $\delta_l = \sup\{|\mathcal{E}(\boldsymbol{x}) - \mathcal{E}(\boldsymbol{y})|; x_i = y_i \forall i \in V \text{ with } i \neq l\}$. Here μ is an arbitrary starting distribution and $\frac{1}{2}|\boldsymbol{\mu} - \boldsymbol{\pi}|$ is the distance in variation as defined in (15).

4 Restricted Boltzmann Machines

A RBM (also denoted as Harmonium [34]) is an MRF associated with a bipartite undirected graph as shown in Fig. 1. It consists of m visible units $\boldsymbol{V} = (V_1, ..., V_m)$ to represent observable data and n hidden units $\boldsymbol{H} = (H_1, ..., H_n)$ to capture dependencies between observed variables. In binary RBMs, our focus in this tutorial, the random variables $(\boldsymbol{V}, \boldsymbol{H})$ take values $(\boldsymbol{v}, \boldsymbol{h}) \in \{0, 1\}^{m+n}$ and the joint probability distribution under the model is given by the Gibbs distribution $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})}$ with the energy function

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \ . \tag{20}$$

For all $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$, w_{ij} is a real valued weight associated with the edge between units V_j and H_i and b_j and c_i are real valued bias terms associated with the jth visible and the ith hidden variable, respectively.

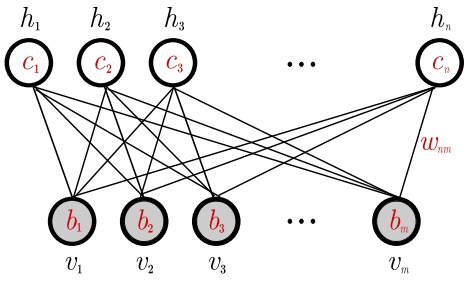

Fig. 1. The undirected graph of an RBM with n hidden and m visible variables

The graph of an RBM has only connections between the layer of hidden and visible variables but not between two variables of the same layer. In terms of probability this means that the hidden variables are independent given the state of the visible variables and vice versa:

$$p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \prod_{i=1}^{n} p(h_i \,|\, \boldsymbol{v}) \text{ and } p(\boldsymbol{v} \,|\, \boldsymbol{h}) = \prod_{i=1}^{m} p(v_i \,|\, \boldsymbol{h}) \ . \tag{21}$$

The absence of connections between hidden variables makes the marginal distribution (7) of the visible variables easy to calculate:

$$\begin{aligned}
p(\boldsymbol{v}) &= \frac{1}{Z} \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \\
&= \frac{1}{Z} \sum_{h_1} \sum_{h_2} \cdots \sum_{h_n} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} e^{h_i \left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)} \\
&= \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \sum_{h_1} e^{h_1 \left(c_1 + \sum_{j=1}^{m} w_{1j} v_j\right)} \sum_{h_2} e^{h_2 \left(c_2 + \sum_{j=1}^{m} w_{2j} v_j\right)} \cdots \sum_{h_n} e^{h_n \left(c_n + \sum_{j=1}^{m} w_{nj} v_j\right)} \\
&= \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} \sum_{h_i} e^{h_i \left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)} \\
&= \frac{1}{Z} \prod_{j=1}^{m} e^{b_j v_j} \prod_{i=1}^{n} \left(1 + e^{c_i + \sum_{j=1}^{m} w_{ij} v_j}\right)
\end{aligned} \tag{22}$$

This equation shows why a (marginalized) RBM can be regarded as a *product of experts* model [15,39], in which a number of "experts" for individual components of the observations are combined multiplicatively.

Any distribution on $\{0,1\}^m$ can be modeled arbitrarily well by an RBM with m visible and $k+1$ hidden units, where k denotes the cardinality of the support set of the target distribution, that is, the number of input elements from $\{0,1\}^m$ that have a non-zero probability of being observed [24]. It has been shown recently that even less units can be sufficient depending on the patterns in the support set [30].

The RBM can be interpreted as a stochastic neural network, where nodes and edges correspond to neurons and synaptic connections, respectively. The conditional probability of a single variable being one can be interpreted as the firing rate of a (stochastic) neuron with sigmoid activation function $\sigma(x) = 1/(1+e^{-x})$, because it holds:

$$p(H_i = 1 \,|\, \boldsymbol{v}) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right) \tag{23}$$

and

$$p(V_j = 1 \,|\, \boldsymbol{h}) = \sigma\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right) \ . \tag{24}$$

To see this, let $\boldsymbol{v}_{-l}$ denote the state of all visible units except the lth one and let us define

$$\alpha_l(\boldsymbol{h}) := -\sum_{i=1}^{n} w_{il} h_i - b_l \tag{25}$$

and

$$\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) := -\sum_{i=1}^{n} \sum_{j=1, j\neq l}^{m} w_{ij} h_i v_j - \sum_{j=1, j\neq l}^{m} b_i v_i - \sum_{i=1}^{n} c_i h_i \ . \tag{26}$$

Then $E(\boldsymbol{v}, \boldsymbol{h}) = \beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) + v_l \alpha_l(\boldsymbol{h})$, where $v_l \alpha_l(\boldsymbol{h})$ collects all terms involving v_l and we can write [2]:

$$\begin{aligned}
p(V_l = 1 \,|\boldsymbol{h}) = p(V_l = 1 \,|\boldsymbol{v}_{-l}, \boldsymbol{h}) &= \frac{p(V_l = 1, \boldsymbol{v}_{-l}, \boldsymbol{h})}{p(\boldsymbol{v}_{-l}, \boldsymbol{h})} \\
= \frac{e^{-E(v_l=1, \boldsymbol{v}_{-l}, \boldsymbol{h})}}{e^{-E(v_l=1, \boldsymbol{v}_{-l}, \boldsymbol{h})} + e^{-E(v_l=0, \boldsymbol{v}_{-l}, \boldsymbol{h})}} &= \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha_l(\boldsymbol{h})} + e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 0 \cdot \alpha_l(\boldsymbol{h})}} \\
= \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})} + e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})}} &= \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot \left(e^{-\alpha_l(\boldsymbol{v}_{-l}, \boldsymbol{h})} + 1\right)} \\
= \frac{e^{-\alpha_l(\boldsymbol{v}_{-l}, \boldsymbol{h})}}{e^{-\alpha_l(\boldsymbol{v}_{-l}, \boldsymbol{h})} + 1} &= \frac{\frac{1}{e^{\alpha_l(\boldsymbol{h})}}}{\frac{1}{e^{\alpha_l(\boldsymbol{h})}} + 1} = \frac{1}{1 + e^{\alpha_l(\boldsymbol{v}_{-l}, \boldsymbol{h})}} \\
&= \sigma(-\alpha_l(\boldsymbol{h})) = \sigma\left(\sum_{i=1}^{n} w_{il} h_i + b_j\right)
\end{aligned} \tag{27}$$

The independence between the variables in one layer makes Gibbs sampling especially easy: Instead of sampling new values for all variables subsequently, the states of all variables in one layer can be sampled jointly. Thus, Gibbs sampling can be performed in just two sub steps: sampling a new state $\boldsymbol{h}$ for the hidden neurons based on $p(\boldsymbol{h}|\boldsymbol{v})$ and sampling a state $\boldsymbol{v}$ for the visible layer based on $p(\boldsymbol{v}|\boldsymbol{h})$. This is also referred to as *block Gibbs sampling*.

As mentioned in the introduction, an RBM can be reinterpreted as a standard feed-forward neural network with one layer of non-linear processing units. From this perspective the RBM is viewed as a deterministic function $\{0,1\}^m \to \mathbb{R}^n$ that maps an input $\boldsymbol{v} \in \{0,1\}^m$ to $\boldsymbol{y} \in \mathbb{R}^n$ with $y_i = p(H_i = 1|\boldsymbol{v})$. That is, an observation is mapped to the expected value of the hidden neurons given the observation.

4.1 The Gradient of the Log-Likelihood

The log-likelihood gradient of an MRF can be written as the sum of two expectations, see (9). For RBMs the first term of (9) (i.e., the expectation of the energy gradient under the conditional distribution of the hidden variables given a training sample $\boldsymbol{v}$) can be computed efficiently because it factorizes nicely. For example, w.r.t. the parameter w_{ij} we get:

$$\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} = \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j$$

$$= \sum_{\boldsymbol{h}} \prod_{k=1}^{n} p(h_k \,|\, \boldsymbol{v}) h_i v_j = \sum_{h_i} \sum_{\boldsymbol{h}_{-i}} p(h_i \,|\, \boldsymbol{v}) p(\boldsymbol{h}_{-i} \,|\, \boldsymbol{v}) h_i v_j$$

$$= \sum_{h_i} p(h_i \,|\, \boldsymbol{v}) h_i v_j \underbrace{\sum_{\boldsymbol{h}_{-i}} p(\boldsymbol{h}_{-i} | \boldsymbol{v})}_{=1} = p(H_i = 1 | \boldsymbol{v}) v_j = \sigma\Bigg(\sum_{j=1}^{m} w_{ij} v_j + c_i \Bigg) v_j \quad (28)$$

Since the second term in (9) can also be written as $\sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}}$ or $\sum_{\boldsymbol{h}} p(\boldsymbol{h}) \sum_{\boldsymbol{v}} p(\boldsymbol{v} \,|\, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}}$ we can also reduce its computational complexity by applying the same kind of factorization to the inner sum, either factorizing over the hidden variables as shown above or factorizing over the visible variables in an analogous way. However, the computation remains intractable for regular sized RBMs because its complexity is still exponential in the size of the smallest layer (the outer sum still runs over either 2^m or 2^n states).

Using the factorization trick (28) the derivative of the log-likelihood of a single training pattern $\boldsymbol{v}$ w.r.t. the weight w_{ij} becomes

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial w_{ij}} = - \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v}, \boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}}$$

$$= \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j$$

$$= p(H_i = 1 | \, \boldsymbol{v}) v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_i = 1 | \, \boldsymbol{v}) v_j \ . \quad (29)$$

For the mean of this derivative over a training set $S = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell\}$ often the following notations are used:

$$\frac{1}{\ell} \sum_{\boldsymbol{v} \in S} \frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial w_{ij}} = \frac{1}{\ell} \sum_{\boldsymbol{v} \in S} \left[-\mathbb{E}_{p(\boldsymbol{h} \,|\, \boldsymbol{v})} \left[\frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} \right] + \mathbb{E}_{p(\boldsymbol{h}, \boldsymbol{v})} \left[\frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} \right] \right]$$

$$= \frac{1}{\ell} \sum_{\boldsymbol{v} \in S} \left[\mathbb{E}_{p(\boldsymbol{h} \,|\, \boldsymbol{v})} \left[v_i h_j \right] - \mathbb{E}_{p(\boldsymbol{h}, \boldsymbol{v})} \left[v_i h_j \right] \right]$$

$$= \left\langle v_i h_j \right\rangle_{p(\boldsymbol{h} \,|\, \boldsymbol{v}) q(\boldsymbol{v})} - \left\langle v_i h_j \right\rangle_{p(\boldsymbol{h}, \boldsymbol{v})} \quad (30)$$

with q denoting the empirical distribution. This gives the often stated rule:

$$\sum_{\boldsymbol{v} \in S} \frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial w_{ij}} \propto \left\langle v_i h_j \right\rangle_{\text{data}} - \left\langle v_i h_j \right\rangle_{\text{model}} \quad (31)$$

Analogously to (29) we get the derivatives w.r.t. the bias parameter b_j of the jth visible variable

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial b_j} = v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) v_j \quad (32)$$

and w.r.t. the bias parameter c_i of the ith hidden variable

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial c_i} = p(H_i = 1 |\, \boldsymbol{v}) - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_i = 1 |\, \boldsymbol{v}) \ . \tag{33}$$

To avoid the exponential complexity of summing over all values of the visible variables (or all values of the hidden if one decides to factorize over the visible variables beforehand) when calculating the second term of the log-likelihood gradient – or the second terms of (29), (32), and (33) – one can approximate this expectation by samples from the model distribution. These samples can be obtained by Gibbs sampling. This requires running the Markov chain "long enough" to ensure convergence to stationarity. Since the computational costs of such an MCMC approach are still too large to yield an efficient learning algorithm common RBM learning techniques – as described in the following section – introduce additional approximations.

5 Approximating the RBM Log-Likelihood Gradient

All common training algorithms for RBMs approximate the log-likelihood gradient given some data and perform gradient ascent on these approximations. Selected learning algorithms will be described in the following section, starting with contrastive divergence learning.

5.1 Contrastive Divergence

Obtaining unbiased estimates of log-likelihood gradient using MCMC methods typically requires many sampling steps. However, recently it was shown that estimates obtained after running the chain for just a few steps can be sufficient for model training [15]. This leads to *contrastive divergence* (CD) learning, which has become a standard way to train RBMs [15,4,18,3,17].

The idea of k-step contrastive divergence learning (CD-k) is quite simple: Instead of approximating the second term in the log-likelihood gradient by a sample from the RBM-distribution (which would require to run a Markov chain until the stationary distribution is reached), a Gibbs chain is run for only k steps (and usually $k = 1$). The Gibbs chain is initialized with a training example $\boldsymbol{v}^{(0)}$ of the training set and yields the sample $\boldsymbol{v}^{(k)}$ after k steps. Each step t consists of sampling $\boldsymbol{h}^{(t)}$ from $p(\boldsymbol{h}|\boldsymbol{v}^{(t)})$ and sampling $\boldsymbol{v}^{(t+1)}$ from $p(\boldsymbol{v}|\boldsymbol{h}^{(t)})$ subsequently. The gradient (equation (9)) w.r.t. $\boldsymbol{\theta}$ of the log-likelihood for one training pattern $\boldsymbol{v}^{(0)}$ is then approximated by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \ . \tag{34}$$

The derivatives in direction of the single parameters are obtained by "estimating" the expectations over $p(\boldsymbol{v})$ in (29), (32) and (33) by the single sample $\boldsymbol{v}^{(k)}$. A batch version of CD-k can be seen in algorithm 1.

Algorithm 1. k-step contrastive divergence

Input: RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S
Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$

```
init Δw_ij = Δb_j = Δc_i = 0 for i = 1,...,n, j = 1,...,m
forall the v ∈ S do
    v^(0) ← v
    for t = 0,...,k−1 do
        for i = 1,...,n do sample h_i^(t) ~ p(h_i | v^(t))
        for j = 1,...,m do sample v_j^(t+1) ~ p(v_j | h^(t))
    for i = 1,...,n, j = 1,...,m do
        Δw_ij ← Δw_ij + p(H_i = 1 | v^(0))·v_j^(0) − p(H_i = 1 | v^(k))·v_j^(k)
        Δb_j ← Δb_j + v_j^(0) − v_j^(k)
        Δc_i ← Δc_i + p(H_i = 1 | v^(0)) − p(H_i = 1 | v^(k))
```

Since $\boldsymbol{v}^{(k)}$ is not a sample from the stationary model distribution the approximation (34) is biased. Obviously, the bias vanishes as $k \to \infty$. That CD is a biased approximation becomes also clear by realizing that it does not maximize the likelihood of the data under the model but the difference of two KL-divergences [15]:

$$\mathrm{KL}(q|p) - \mathrm{KL}(p_k|p) \ , \tag{35}$$

where q is the empirical distribution and p_k is the distribution of the visible variables after k steps of the Markov chain. If the chain already reached stationarity it holds $p_k = p$ and thus $\mathrm{KL}(p_k|p) = 0$ and the approximation error of CD vanishes.

The theoretical results from [3] give a good understanding of the CD approximation and the corresponding bias by showing that the log-likelihood gradient can – based on a Markov chain – be expressed as a sum of terms containing the k-th sample:

Theorem 1 (Bengio and Delalleau [3]). *For a converging Gibbs chain*

$$\boldsymbol{v}^{(0)} \Rightarrow \boldsymbol{h}^{(0)} \Rightarrow \boldsymbol{v}^{(1)} \Rightarrow \boldsymbol{h}^{(1)} \ldots$$

starting at data point $\boldsymbol{v}^{(0)}$, the log-likelihood gradient can be written as

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p(\boldsymbol{v}^{(0)}) = &- \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \\ &+ E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \right] + E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial \boldsymbol{\theta}} \right] \end{aligned} \tag{36}$$

and the final term converges to zero as k goes to infinity.

The first two terms in equation (36) just correspond to the expectation of the CD approximation (under p_k) and the bias is given by the final term.

The approximation error does not only depend on the value of k but also on the rate of convergence or the mixing rate of the Gibbs chain. The rate describes how fast the Markov chain approaches the stationary distribution. The mixing rate of the Gibbs chain of an RBM is up to the magnitude of the model parameters [15,7,3,12]. This becomes clear by considering that the conditional probabilities $p(v_j|\boldsymbol{h})$ and $p(h_i|\boldsymbol{v})$ are given by thresholding $\sum_{i=1}^{n} w_{ij}h_i + b_j$ and $\sum_{j=1}^{m} w_{ij}v_j + c_i$, respectively. If the absolute values of the parameters are high, the conditional probabilities can get close to one or zero. If this happens, the states get more and more "predictable" and the Markov chain changes its state slowly. An empirical analysis of the dependency between the size of the bias and magnitude of the parameters can be found in [3].

An upper bound on the expectation of the CD approximation error under the empirical distribution is given by the following theorem [12]:

Theorem 2 (Fischer and Igel [12]). *Let p denote the marginal distribution of the visible units of an RBM and let q be the empirical distribution defined by a set of samples $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell$. Then an upper bound on the expectation of the error of the CD-k approximation of the log-likelihood derivative w.r.t some RBM parameter θ_a is given by*

$$\left| E_{q(\boldsymbol{v}^{(0)})} \left[E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] \right] \right| \leq \frac{1}{2} \|q - p\| \left(1 - e^{-(m+n)\Delta} \right)^k \tag{37}$$

with

$$\Delta = \max \left\{ \max_{l \in \{1,\ldots,m\}} \vartheta_l, \max_{l \in \{1,\ldots,n\}} \xi_l \right\} ,$$

where

$$\vartheta_l = \max \left\{ \left| \sum_{i=1}^{n} I_{\{w_{il}>0\}} w_{il} + b_l \right| , \left| \sum_{i=1}^{n} I_{\{w_{il}<0\}} w_{il} + b_l \right| \right\}$$

and

$$\xi_l = \max \left\{ \left| \sum_{j=1}^{m} I_{\{w_{lj}>0\}} w_{lj} + c_l \right| , \left| \sum_{j=1}^{m} I_{\{w_{lj}<0\}} w_{lj} + c_l \right| \right\} .$$

The bound (and probably also the bias) depends on the absolute values of the RBM parameters, on the size of the RBM (the number of variables in the graph), and on the distance in variation between the modeled distribution and the starting distribution of the Gibbs chain.

As a consequence of the approximation error CD-learning does not necessarily lead to a maximum likelihood estimate of the model parameters. Yuille [42] specifies conditions under which CD learning is guaranteed to converge to the maximum likelihood solution, which need not hold for RBM training in general. Examples of energy functions and Markov chains for which CD-1 learning does not converge are given in [27]. The empirical comparisons of the CD-approximation and the true gradient for RBMs small enough that the gradient

is still tractable conducted in [7] and [3] shows that the bias can lead to a convergence to parameters that do not reach the maximum likelihood.

The bias, however, can also lead to a distortion of the learning process: After some learning iterations the likelihood can start to diverge (see figure 2) in the sense that the model systematically gets worse if k is not large [11]. This is especially bad because the log-likelihood is not tractable in reasonable sized RBMs, and so the misbehavior can not be displayed and used as a stopping criterion. Because the effect depends on the magnitude of the weights, weight decay can help to prevent it. However, the weight decay parameter λ, see equation (6), is difficult to tune. If it is too small, weight decay has no effect. If it is too large, the learning converges to models with low likelihood [11].

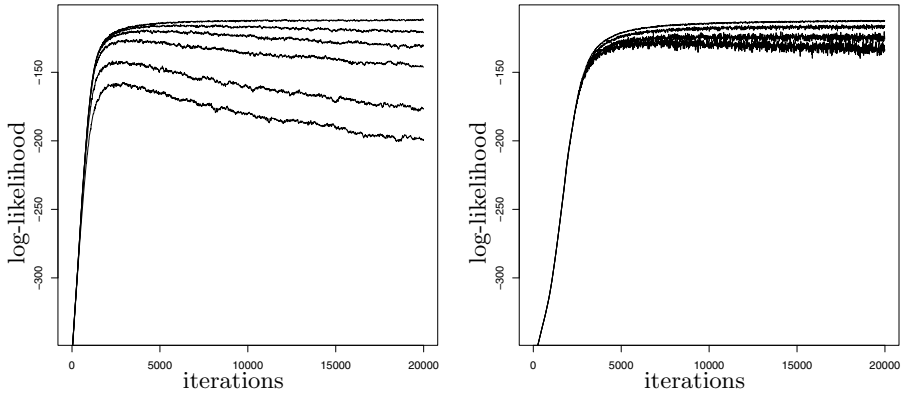

Fig. 2. Evolution of the log-likelihood during batch training of an RBM. In the left plot, CD-k with different values for k (from bottom to top $k = 1, 2, 5, 10, 20, 100$) was used. In the right plot, we employed parallel tempering (PT, see section 5.3) with different numbers M of temperatures (from bottom to top $M = 4, 5, 10, 50$). In PT the inverse temperatures were equally distributed in the interval $[0, 1]$, which may not be the optimal [9]. The training set was given by a 4×4 variant of the Bars-and-Stripes benchmark problem [28]. The learning rate was $\eta = 0.1$ for CD and $\eta = 0.05$ for PT and neither weight decay nor a momentum term were used ($\lambda = \nu = 0$). Shown are medians over 25 runs.

More recently proposed learning algorithms try to yield better approximations of the log-likelihood gradient by sampling from Markov chains with increased mixing rate.

5.2 Persistent Contrastive Divergence

The idea of *persistent contrastive divergence* (PCD, [36]) is described in [41] for log-likelihood maximization of general MRFs and is applied to RBMs in [36]. The PCD approximation is obtained from the CD approximation (34) by replacing the sample $v^{(k)}$ by a sample from a Gibbs chain that is independent

from the sample $v^{(0)}$ of the training distribution. The algorithm corresponds to standard CD learning without reinitializing the visible units of the Markov chain with a training sample each time we want to draw a sample $v^{(k)}$ approximately from the RBM distribution. Instead one keeps "persistent" chains which are run for k Gibbs steps after each parameter update (i.e., the initial state of the current Gibbs chain is equal to $v^{(k)}$ from the previous update step). The fundamental idea underlying PCD is that one could assume that the chains stay close to the stationary distribution if the learning rate is sufficiently small and thus the model changes only slightly between parameter updates [41,36]. The number of persistent chains used for sampling (or the number of samples used to approximate the second term of gradient (9)) is a hyper parameter of the algorithm. In the canonical from, there exists one Markov chain per training example in a batch.

The PCD algorithm was further refined in a variant called *fast persistent contrastive divergence* (FPCD, [37]). Fast PCD tries to reach faster mixing of the Gibbs chain by introducing additional parameters w_{ij}^f, b_j^f, c_i^f (for $i = 1, \ldots, n$ and $j = 1, \ldots, m$) referred to as *fast* parameters. These new set of parameters is only used for sampling and not in the model itself. When calculating the conditional distributions for Gibbs sampling, the regular parameters are replaced by the sum of the regular and the fast parameters, i.e., Gibbs sampling is based on the probabilities $\tilde{p}(H_i = 1 \,|\, \boldsymbol{v}) = \sigma\left(\sum_{j=1}^{m} (w_{ij} + w_{ij}^f) v_j + (c_i + c_i^f)\right)$ and $\tilde{p}(V_j = 1 \,|\, \boldsymbol{h}) = \sigma\left(\sum_{i=1}^{n} (w_{ij} + w_{ij}^f) h_i + (b_j + b_j^f)\right)$ instead of the conditional probabilities given by (23) and (24). The learning update rule for the fast parameters equals the one for the regular parameters, but with an independent, large learning rate leading to faster changes as well as a large weight decay parameter. Weight decay can also be used for the regular parameters, but it was suggested that regularizing just the fast weights is sufficient [37].

Neither PCD nor FPCD seem to enlarge the mixing rate (or decrease the bias of the approximation) sufficiently to avoid the divergence problem as can be seen in the empirical analysis in [11].

5.3 Parallel Tempering

One of the most promising sampling technique used for RBM-training so far is *parallel tempering* (PT, [33,10,8]). It introduces supplementary Gibbs chains that sample form more and more smoothed replicas of the original distribution. This can be formalized in the following way: Given an ordered set of M temperatures $T_1, T_2, \ldots, T_M$ with $1 = T_1 < T_2 < \cdots < T_M$, we define a set of M Markov chains with stationary distributions

$$p_r(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z_r} e^{-\frac{1}{T_r} E(\boldsymbol{v}, \boldsymbol{h})} \tag{38}$$

for $r = 1, \ldots, M$, where $Z_r = \sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-\frac{1}{T_r} E(\boldsymbol{v}, \boldsymbol{h})}$ is the corresponding partition function, and p_1 is exactly the model distribution.

Algorithm 2. k-step parallel tempering with M temperatures

Input: RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S, current state $\boldsymbol{v}_r$ of Markov chain with stationary distribution p_r for $r = 1, \ldots, M$
Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$

init $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$
forall the $\boldsymbol{v} \in S$ **do**
  for $r = 1, \ldots, M$ **do**
    $\boldsymbol{v}_r^{(0)} \leftarrow \boldsymbol{v}_r$
    for $i = 1, \ldots, n$ **do** sample $h_{r,i}^{(0)} \sim p(h_{r,i} \,|\, \boldsymbol{v}_r^{(0)})$
    for $t = 0, \ldots, k-1$ **do**
      for $j = 1, \ldots, m$ **do** sample $v_{r,j}^{(t+1)} \sim p(v_{r,j} \,|\, \boldsymbol{h}_r^{(t)})$
      for $i = 1, \ldots, n$ **do** sample $h_{r,i}^{(t+1)} \sim p(h_{r,i} \,|\, \boldsymbol{v}_r^{(t+1)})$
    $\boldsymbol{v}_r \leftarrow \boldsymbol{v}_r^{(k)}$
  `/* swapping order below works well in practice [26] */`
  for $r \in \{s \,|\, 2 \leq s \leq M \text{ and } s \bmod 2 = 0\}$ **do**
    swap $(\boldsymbol{v}_r^{(k)}, \boldsymbol{h}_r^{(k)})$ and $(\boldsymbol{v}_{r-1}^{(k)}, \boldsymbol{h}_{r-1}^{(k)})$ with probability given by (40)
  for $r \in \{s \,|\, 3 \leq s \leq M \text{ and } s \bmod 2 = 1\}$ **do**
    swap $(\boldsymbol{v}_r^{k}, \boldsymbol{h}_r^{k})$ and $(\boldsymbol{v}_{r-1}^{k}, \boldsymbol{h}_{r-1}^{k})$ with probability given by (40)
  for $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**
    $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 \,|\, \boldsymbol{v}) \cdot v_j - p(H_i = 1 \,|\, \boldsymbol{v}_1^{(k)}) \cdot v_{1,j}^{(k)}$
    $\Delta b_j \leftarrow \Delta b_j + v_j - v_{1,j}^{(k)}$
    $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 \,|\, \boldsymbol{v}) - p(H_i = 1 \,|\, \boldsymbol{v}_1^{(k)})$

In each step of the algorithm, we run k (usually $k = 1$) Gibbs sampling steps in each tempered Markov chain yielding samples $(\boldsymbol{v}_1, \boldsymbol{h}_1), \ldots, (\boldsymbol{v}_M, \boldsymbol{h}_M)$. After this, two neighboring Gibbs chains with temperatures T_r and T_{r-1} may exchange particles $(\boldsymbol{v}_r, \boldsymbol{h}_r)$ and $(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1})$ with an exchange probability based on the Metropolis ratio,

$$\min\left\{1, \frac{p_r\big(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1}\big) p_{r-1}\big(\boldsymbol{v}_r, \boldsymbol{h}_r\big)}{p_r\big(\boldsymbol{v}_r, \boldsymbol{h}_r\big) p_{r-1}\big(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1}\big)}\right\} , \tag{39}$$

which gives for RBMs

$$\min\left\{1, \exp\left(\left(\frac{1}{T_r} - \frac{1}{T_{r-1}}\right) * \left(E(\boldsymbol{v}_r, \boldsymbol{h}_r) - E(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1})\right)\right)\right\} . \tag{40}$$

After performing these swaps between chains, which enlarge the mixing rate, we take the (eventually exchanged) sample $\boldsymbol{v}_1$ of original chain (with temperature $T_1 = 1$) as a sample from the model distribution. This procedure is repeated L times yielding samples $\boldsymbol{v}_{1,1}, \ldots, \boldsymbol{v}_{1,L}$ used for the approximation of the expectation under the RBM distribution in the log-likelihood gradient (i.e., for the

approximation of the second term in (9)). Usually L is set to the number of samples in the (mini) batch of training data as shown in algorithm 2.

Compared to CD, PT introduces computational overhead, but results in a faster mixing Markov chain and thus a less biased gradient approximation. The evolution of the log-likelihood during training using PT with different values of M can be seen in figure 2.

6 RBMs with Real-Valued Variables

So far, we considered only observations represented by binary vectors, but often one would like to model distributions over continuous data. There are several ways to define RBMs with real-valued visible units. As demonstrated by [18], one can model a continuous distribution with a binary RBM by a simple "trick". The input data is scaled to the interval $[0, 1]$ and modeled by the probability of the visible variables to be one. That is, instead of sampling binary values, the expectation $p(V_j = 1|\boldsymbol{h})$ is regarded as the current state of the variable V_j. Except for the continuous values of the visible variables and the resulting changes in the sampling procedure the learning process remains the same. By keeping the energy function as given in (20) and just replacing the state space $\{0, 1\}^m$ of $\boldsymbol{V}$ by $[0, 1]^m$, the conditional distributions of the visible variables belong to the class of truncated exponential distributions. This can be shown in the same way as the sigmoid function for binary RBMs is derived in (27). Visible neurons with a Gaussian distributed conditional are for example gained by augmenting the energy with quadratical terms $\sum_j d_j v_j^2$ weighted by parameters d_j, $j = 1, \ldots, m$.

In contrast to the universal approximation capabilities of standard RBMs on $\{0, 1\}^m$, the subset of real-valued distributions that can be modeled by an RBM with real-valued visible units is rather constrained [38].

More generally, it is possible to cover continuous valued variables by extending the definition of an RBM to any MRF whose energy function is such that $p(\boldsymbol{h}|\boldsymbol{v}) = \prod_i p(h_i|\boldsymbol{v})$ and $p(\boldsymbol{v}|\boldsymbol{h}) = \prod_j p(v_j|\boldsymbol{h})$. As follows directly from the Hammersley-Clifford theorem and as also discussed in [18], this holds for any energy function of the form

$$E(\boldsymbol{v}, \boldsymbol{h}) = \sum_{i,j} \phi_{i,j}(h_i, v_j) + \sum_j \omega_j(v_j) + \sum_i \nu_i(h_i)) \tag{41}$$

with real-valued functions $\phi_{i,j}$, ω_j, and ν_i ,$i = 1, \ldots, n$ and $j = 1, \ldots, m$, fulfilling the constraint that the partition function Z is finite. Welling et al. [40] come to almost the same generalized form of the energy function in their framework for constructing *exponential family harmoniums* from arbitrary marginal distributions $p(v_j)$ and $p(h_i)$ from the exponential family.

7 Loosening the Restrictions

In this closing section, we will give a very brief outlook on selected extensions of RBMs that loosen the imposed restrictions on the bipartite network topology

by introducing dependencies on further random variables or by allowing for arbitrary connections between nodes in the model.

Conditional RBMs. Several generalizations and extensions of RBMs exist. A notable example are *conditional RBMs* (e.g., example [35,29]). In these models, some of the parameters in the RBM energy are replaced by parametrized functions of some conditioning random variables, see [2] for an introduction.

Boltzmann Machines. Removing the "R" from the RBM brings us back to where everything started, to the general Boltzmann machine [1]. These are MRFs consisting of a set of hidden and visible variables where the energy is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = \\ -\sum_{i=1}^{n}\sum_{j=1}^{m} h_i w_{ij} v_j - \sum_{k=1}^{m}\sum_{l<k} v_k u_{kl} v_l - \sum_{k=1}^{n}\sum_{l<k} h_k y_{kl} h_l - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \ . \tag{42}$$

The graph corresponds to the one of an RBM with additional connections between the variables of one layer. These dependencies make sampling more complex (in Gibbs sampling each variable has to be updated independently) and thus training more difficult. However, specialized learning algorithms for particular "deep" graph structures have been developed [32].

8 Next Steps

The goal of this tutorial was to introduce RBMs from the probabilistic graphical model perspective. The text is meant to supplement existing tutorials, and it is biased in the sense that it focuses on material that we found helpful in our work. We hope that the reader is now equipped to move on to advanced models building on RBMs – in particular to deep learning architectures, where [2] may serve as an excellent starting point.

All experiments in this tutorial can be reproduced using the open source machine learning library Shark [20], which implements most of the models and algorithms that were discussed.

Acknowledgments. The authors acknowledge support from the German Federal Ministry of Education and Research within the National Network Computational Neuroscience under grant number 01GQ0951 (Bernstein Fokus "Learning behavioral models: From human experiment to technical assistance").

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science 9, 147–169 (1985)
2. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 21(6), 1601–1621 (2009)
3. Bengio, Y., Delalleau, O.: Justifying and generalizing contrastive divergence. Neural Computation 21(6), 1601–1621 (2009)

4. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., Montreal, U.: Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing (NIPS 19), pp. 153–160. MIT Press (2007)
5. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
6. Brémaud, P.: Markov chains: Gibbs fields, Monte Carlo simulation, and queues. Springer (1999)
7. Carreira-Perpiñán, M.Á., Hinton, G.E.: On contrastive divergence learning. In: 10th International Workshop on Artificial Intelligence and Statistics (AISTATS 2005), pp. 59–66 (2005)
8. Cho, K., Raiko, T., Ilin, A.: Parallel tempering is efficient for learning restricted Boltzmann machines. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010), pp. 3246–3253. IEEE Press (2010)
9. Desjardins, G., Courville, A., Bengio, Y.: Adaptive parallel tempering for stochastic maximum likelihood learning of RBMs. In: Lee, H., Ranzato, M., Bengio, Y., Hinton, G., LeCun, Y., Ng, A.Y. (eds.) NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning (2010)
10. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Dellaleau, O.: Parallel tempering for training of restricted Boltzmann machines. In: JMLR Workshop and Conference Proceedings: AISTATS 2010, vol. 9, pp. 145–152 (2010)
11. Fischer, A., Igel, C.: Empirical Analysis of the Divergence of Gibbs Sampling Based Learning Algorithms for Restricted Boltzmann Machines. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part III. LNCS, vol. 6354, pp. 208–217. Springer, Heidelberg (2010)
12. Fischer, A., Igel, C.: Bounding the bias of contrastive divergence learning. Neural Computation 23, 664–673 (2011)
13. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6, 721–741 (1984)
14. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57(1), 97–109 (1970)
15. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771–1800 (2002)
16. Hinton, G.E.: Boltzmann machine. Scholarpedia 2(5), 1668 (2007)
17. Hinton, G.E.: Learning multiple layers of representation. Trends in Cognitive Sciences 11(10), 428–434 (2007)
18. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)
19. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
20. Igel, C., Glasmachers, T., Heidrich-Meisner, V.: Shark. Journal of Machine Learning Research 9, 993–996 (2008)
21. Kivinen, J., Williams, C.: Multiple texture boltzmann machines. In: JMLR Workshop and Conference Proceedings: AISTATS 2012, vol. 22, pp. 638–646 (2012)
22. Koller, D., Friedman, N.: Probabilistic graphical models: Principles and techniques. MIT Press (2009)
23. Lauritzen, S.L.: Graphical models. Oxford University Press (1996)
24. Le Roux, N., Bengio, Y.: Representational power of restricted Boltzmann machines and deep belief networks. Neural Computation 20(6), 1631–1649 (2008)
25. Le Roux, N., Heess, N., Shotton, J., Winn, J.M.: Learning a generative model of images by factoring appearance and shape. Neural Computation 23(3), 593–650 (2011)

26. Lingenheil, M., Denschlag, R., Mathias, G., Tavan, P.: Efficiency of exchange schemes in replica exchange. Chemical Physics Letters 478, 80–84 (2009)
27. MacKay, D.J.C.: Failures of the one-step learning algorithm. Cavendish Laboratory, Madingley Road, Cambridge CB3 0HE, UK (2001), http://www.cs.toronto.edu/~mackay/gbm.pdf
28. MacKay, D.J.C.: Information Theory, Inference & Learning Algorithms. Cambridge University Press (2002)
29. Mnih, V., Larochelle, H., Hinton, G.: Conditional restricted Boltzmann machines for structured output prediction. In: Cozman, F.G., Pfeffer, A. (eds.) Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2011), p. 514. AUAI Press (2011)
30. Montufar, G., Ay, N.: Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines. Neural Comput. 23(5), 1306–1319
31. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations, pp. 318–362. MIT Press (1986)
32. Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: JMLR Workshop and Conference Proceedings: AISTATS 2009, vol. 5, pp. 448–455 (2009)
33. Salakhutdinov, R.: Learning in Markov random fields using tempered transitions. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) Advances in Neural Information Processing Systems 22, pp. 1598–1606 (2009)
34. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. In: Rumelhart, D.E., McClelland, J.L. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations, pp. 194–281. MIT Press (1986)
35. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems (NIPS 19), pp. 1345–1352. MIT Press (2007)
36. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) International Conference on Machine learning (ICML), pp. 1064–1071. ACM (2008)
37. Tieleman, T., Hinton, G.E.: Using fast weights to improve persistent contrastive divergence. In: Pohoreckyj Danyluk, A., Bottou, L., Littman, M.L. (eds.) International Conference on Machine Learning (ICML), pp. 1033–1040. ACM (2009)
38. Wang, N., Melchior, J., Wiskott, L.: An analysis of Gaussian-binary restricted Boltzmann machines for natural images. In: Verleysen, M. (ed.) European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), pp. 287–292. d-side publications, Evere (2012)
39. Welling, M.: Product of experts. Scholarpedia 2(10), 3879 (2007)
40. Welling, M., Rosen-Zvi, M., Hinton, G.: Exponential family harmoniums with an application to information retrieval. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems (NIPS 17), pp. 1481–1488. MIT Press, Cambridge (2005)
41. Younes, L.: Maximum likelihood estimation of Gibbs fields. In: Possolo, A. (ed.) Proceedings of an AMS-IMS-SIAM Joint Conference on Spacial Statistics and Imaging. Lecture Notes Monograph Series, Institute of Mathematical Statistics, Hayward (1991)
42. Yuille, A.L.: The convergence of contrastive divergence. In: Saul, L., Weiss, Y., Bottou, L. (eds.) Advances in Neural Processing Systems (NIPS 17), pp. 1593–1600. MIT Press (2005)