Training Restricted Boltzmann Machines: An Introduction*

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

[1] Institut für Neuroinformatik, Ruhr-Universität Bochum
Universitätsstraße 150, 44780 Bochum, Germany
[2] Department of Computer Science, University of Copenhagen
Universitetsparken 5, 2100 Copenhagen Ø, Denmark

Abstract. Restricted Boltzmann machines (RBMs) are probabilistic graphical models that can be interpreted as stochastic neural networks. They have attracted much attention as building blocks for the multi-layer learning systems called deep belief networks, and variants and extensions of RBMs have found application in a wide range of pattern recognition tasks. This tutorial introduces RBMs from the viewpoint of Markov random fields, starting with the required concepts of undirected graphical models. Different learning algorithms for RBMs, including contrastive divergence learning and parallel tempering, are discussed. As sampling from RBMs, and therefore also most of their learning algorithms, are based on Markov chain Monte Carlo (MCMC) methods, an introduction to Markov chains and MCMC techniques is provided. Experiments demonstrate relevant aspects of RBM training.

Keywords: Restricted Boltzmann machines, Markov random fields, Markov chains, Gibbs sampling, neural networks, contrastive divergence learning, parallel tempering

1 Introduction

In the last years, models extending or borrowing concepts from *restricted Boltzmann machines* (RBMs, [49]) have enjoyed much popularity for pattern analysis and generation, with applications including image classification, processing, and generation [25, 51, 34, 28, 48, 31]; learning movement patterns [52, 53]; collaborative filtering for movie recommendations [47]; extraction of semantic document representations [46, 17, 60]; and acoustic modeling [40]. As the name implies, RBMs are a special case of general Boltzmann machines. The latter were introduced as bidirectionally connected networks of stochastic processing units, which can be interpreted as neural network models [1, 22]. A Boltzmann machine is a parameterized model representing a probability distribution, and it can be used to learn important aspects of an unknown *target distribution* based on samples from this target distribution. These samples, or observations, are referred to as the training data. Learning or training a Boltzmann machine means adjusting its parameters such that the probability distribution the machine represents fits the training data as well as possible.

In general, learning a Boltzmann machine is computationally demanding. However, the learning problem can be simplified by imposing restrictions on the network topology, which leads us to RBMs, the topic of this tutorial. In Boltzmann machines two types of units can be distinguished. They have *visible neurons* and potentially *hidden neurons*. Restricted Boltzmann machines always have both types of units, and these can be thought of as being arranged in two layers, see Fig. 1 for an illustration. 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 the pixels in the images) and can be viewed as non-linear feature detectors [22]. In the RBMs network graph, each neuron is connected to all the neurons in the other layer. However, there are no connections between neurons in the same layer, and this restriction gives the RBM its name.

* The final version of this manuscript has been published as: Asja Fischer and Christian Igel. Training Restricted Boltzmann Machines: An Introduction. *Pattern Recognition* 47:25-39, 2014

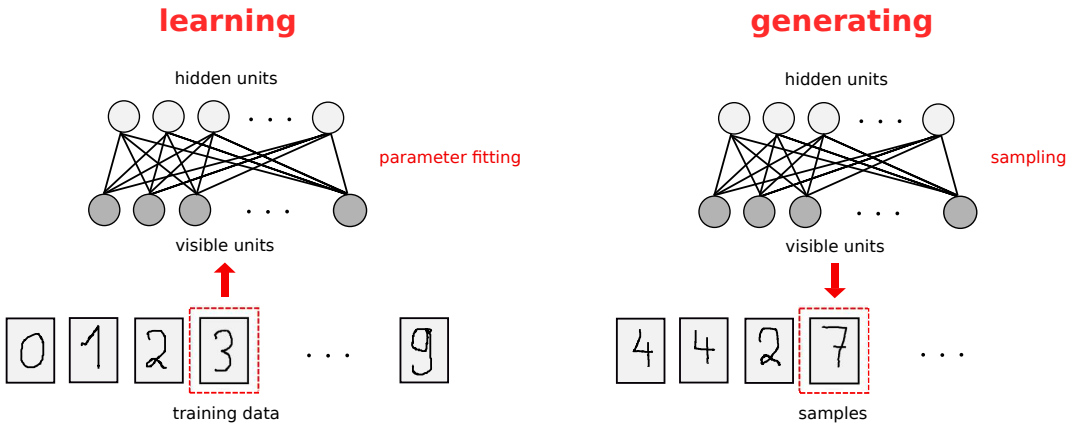

Fig. 1. Left: Learning an RBM corresponds to fitting its parameters such that the distribution represented by the RBM models the distribution underlying the training data, here handwritten digits. Right: After learning, the trained RBM can be used to generate samples from the learned distribution.

Now, what is learning RBMs good for? After successful learning, an RBM provides a closed-form representation of the distribution underlying the training data. It is a generative model that allows sampling from the learned distribution (e.g., to generate image textures [34, 28]), in particular from the marginal distributions of interest, see right plot of Fig. 1. For example, we can fix some visible units corresponding to a partial observation (i.e., we set the corresponding visible variables to the observed values and treat them as constants) and sample the remaining visible units to complete the observation, for example, to solve an image inpainting task [28, 51], see Fig. 7 in Sec. 7. In this way, RBMs can also be used as classifiers: The RBM is trained to model the joint probability distribution of inputs (explanatory variables) and the corresponding labels (response/output variables), both represented by the visible units of the RBM. This is illustrated in the left plot of Fig. 2. Afterwards, a new input pattern can be clamped to the corresponding visible variables and the label can be predicted by sampling, as shown in the right plot of Fig. 2).

Compared to the 1980s when RBMs were first introduced [49], they can now be applied to more interesting problems due to the increase in computational power and the development of new learning strategies [21]. Restricted Boltzmann machines have received a lot of attention recently after being proposed as the building blocks for the multi-layer learning architectures called deep belief networks (DBNs, [25, 23]). The basic idea underlying these deep architectures is that the hidden neurons of a trained RBM represent relevant features of the observations, and that these features can serve as input for another RBM, see Fig. 3 for an illustration. 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. When viewed as neural networks they are used as functions mapping the observations to the expectations of the latent variables in the top layer. These can be interpreted as the learned features, which can, for example, serve as inputs for a supervised learning system. Furthermore, the neural network corresponding to a trained RBM or DBN can be augmented by an output layer where the additional units represent labels (e.g., corresponding to classes) of the observations. Then we have a standard neural network for classification or regression that can be further trained by standard supervised learning algorithms [43]. 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 some of the problems that have been observed when training multi-layer neural networks [25].

Boltzmann machines can be regarded as probabilistic graphical models, namely undirected graphical models also known as Markov random fields (MRFs) [29]. The embedding into the framework of probabilistic graphical models provides immediate access to a wealth of theoretical results and well-

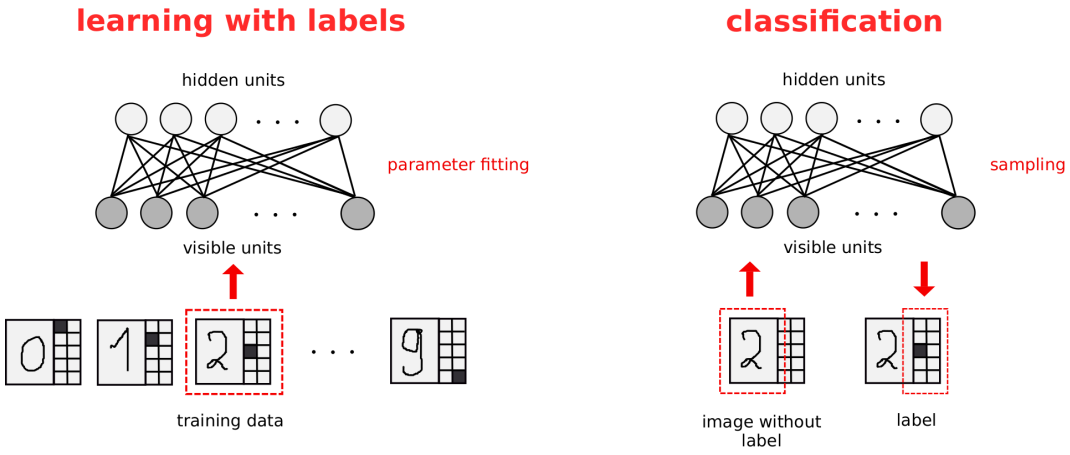

Fig. 2. Left: RBM trained on labeled data, here images of handwritten digits combined with ten binary indicator variables, one of which is set to 1 indicating that the image shows a particular digit while the others are set to 0. Right: The label corresponding to an input image is obtained by fixing the visible variables corresponding to the image and then sampling the remaining visible variables corresponding to the labels from the (marginalized) joined probability distribution of images and labels modeled by the RBM.

developed algorithms. Therefore, we introduce RBMs from this perspective after providing the required background on MRFs. This approach and the coverage of more recent learning algorithms and theoretical results distinguishes this tutorial from others. Section 2 will provide the introduction to MRFs and unsupervised MRF learning. Training of RBMs (i.e., the fitting of the parameters) is usually based on gradient-based maximization of the likelihood of the RBM parameters given the training data, that is, the probability that the distribution modeled by the RBM generated the data. Computing the likelihood of an undirected graphical model or its gradient is in general computationally intensive, and this also holds for RBMs. Thus, sampling-based methods are employed to approximate the likelihood 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. These methods will be presented along with the basic concepts of Markov chain theory in Sec. 3. Then RBMs will be formally described in Sec. 4, and the application of MCMC algorithms to RBMs training will be the topic of Sec. 5. Finally, we will discuss RBMs with real-valued variables before concluding with some experiments.

2 Graphical models

Probabilistic graphical models describe probability distributions by mapping conditional dependence and independence properties between random variables using a graph structure. Two sets of random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ are conditionally independent given a set of random variables $\boldsymbol{Z}$ if for all values $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ of the variables, we have $p(\boldsymbol{x}, \boldsymbol{y} \,|\, \boldsymbol{z}) = p(\boldsymbol{x} \,|\, \boldsymbol{z})p(\boldsymbol{y} \,|\, \boldsymbol{z})$, which implies $p(\boldsymbol{x} \,|\, \boldsymbol{y}, \boldsymbol{z}) = p(\boldsymbol{x} \,|\, \boldsymbol{z})$ and $p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{y} \,|\, \boldsymbol{z})$. 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 kinds 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 last. 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 [29].

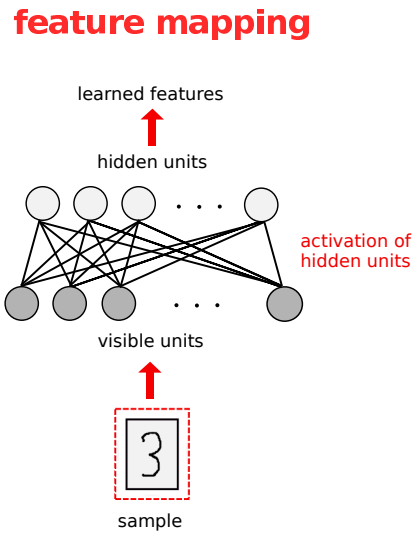

Fig. 3. The trained RBM can be used as a feature extractor. An input pattern is clamped to the visible neurons. The conditional probabilities of the hidden neurons to be 1 are interpreted as a new representation of the input. This new representation can serve as input to another RBM or to a different learning system.

2.1 Undirected graphs and Markov random fields

First, we will summarize some fundamental concepts from graph theory. An *undirected graph* is an ordered pair $G = (V, E)$, where V is a finite set of nodes and E is a set of undirected edges. An edge consists 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. An example of an undirected graph can be seen in Fig. 4. Here the neighborhood of node v_4 is $\{v_2, v_5, v_6\}$.

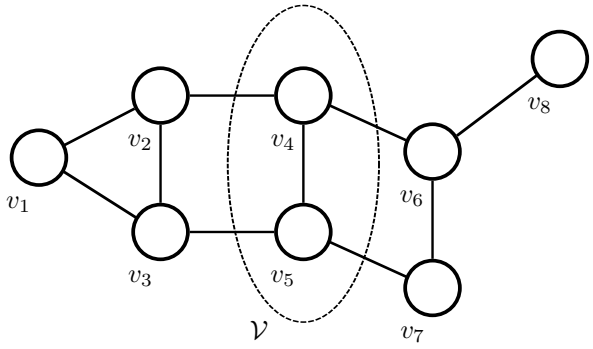

Fig. 4. An example of an undirected graph. The nodes v_1 and v_8 are separated by $\mathcal{V} = \{v_4, v_5\}$.

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 undirected graph in Fig. 4, both $\{v_1, v_2\}$ and $\{v_1, v_2, v_3\}$ are cliques but only the latter is maximal. 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 $\mathcal{V}$. For an illustration of this concept, see Fig. 4).

Given an undirected graph $G = (V, E)$, we now associate, to each node $v \in V$, a random variable X_v taking values in a state space Λ_v. To ease the notation, we assume $\Lambda_v = \Lambda$ for all $v \in V$. The set of random variables $\boldsymbol{X} = (X_v)_{v \in V}$ is called a *Markov random field* (MRF) if the joint probability distribution p fulfills the *(local) Markov property* w.r.t. the graph. This property is fulfilled if for all $v \in V$ the random variable X_v is conditionally independent of all other variables given its neighborhood $(X_w)_{w \in \mathcal{N}_v}$. That is, for all $v \in V$ and all $\boldsymbol{x} \in \Lambda^{|V|}$, one has that $p(x_v \,|\, (x_w)_{w \in V \setminus \{v\}}) = p(x_v \,|\, (x_w)_{w \in \mathcal{N}_v})$.

There exist two other types of Markov properties, which are equivalent to the local Markov property if the probability distribution of the MRF is strictly positive. The MRF is said to have the *global Markov property* if for any three disjunct subsets $\mathcal{A}, \mathcal{B}, \mathcal{S} \subset V$, such that 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 conditionally independent given $(X_s)_{s \in \mathcal{S}}$, i.e., for all $\boldsymbol{x} \in \Lambda^{|V|}$ one has that $p\left((x_a)_{a \in \mathcal{A}} \,|\, (x_t)_{t \in \mathcal{S} \cup \mathcal{B}}\right) = p\left((x_a)_{a \in \mathcal{A}} \,|\, (x_t)_{t \in \mathcal{S}}\right)$. The *pairwise Markov property* means that any two non-adjacent variables are conditionally independent given all other variables: if $\{v, w\} \notin E$, then $p(x_v, x_w \,|\, (x_t)_{t \in V \setminus \{v,w\}}) = p(x_v \,|\, (x_t)_{t \in V \setminus \{v,w\}}) p(x_w \,|\, (x_t)_{t \in V \setminus \{v,w\}})$ for all $\boldsymbol{x} \in \Lambda^{|V|}$.

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 of MRF distributions. An answer to this question is given by the Hammersley–Clifford Theorem (for rigorous formulations and proofs we refer to [32, 29]):

Theorem 1. *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 over 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*, satisfying

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

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 the *partition function*.

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

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

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 (if it is strictly positive) be expressed in the form given by (3), which is also called the *Gibbs distribution*.

2.2 Unsupervised MRF learning

Unsupervised learning means learning (important aspects of) an unknown distribution q based on sample data. This includes finding new representations of the data that foster learning, generalization, and communication. If we assume that the structure of the graphical model is known and that 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 approximations 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 it is 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 by 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:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \underbrace{\eta \frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} \Big(\ln \mathcal{L}(\boldsymbol{\theta}^{(t)} \,|\, S) \Big) - \lambda \boldsymbol{\theta}^{(t)} + \nu \Delta \boldsymbol{\theta}^{(t-1)}}_{= \Delta \boldsymbol{\theta}^{(t)}} \tag{6}$$

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 is called *weight decay*, and 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 is seen in feed-forward neural network training [43].

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 the 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 describing 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

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

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 the 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

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

In the last step we used that the conditional probability can be written

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

Note that the last expression of (9) is the difference between 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 "complicated" probability distributions such as 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 [7] and the aforementioned textbooks [5, 29]. An emphasis will be put on Gibbs sampling as an MCMC technique often used for MRF training and in particular for training RBMs.

3.1 Markov chains

A *Markov chain* is a time discrete stochastic process, where the next state of the system depends only on the current state and not on the sequence of events that preceded it. Formally, a *Markov chain* is a family of random variables $X = \{X^{(k)} \,|\, k \in \mathbb{N}_0\}$ taking values in a (in the following considerations, finite) set Ω and for which $\forall k \geq 0$ and $\forall j, i, i_0, \ldots, i_{k-1} \in \Omega$ one has

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

The "memorylessness" of a stochastic process expressed by (11) is also referred to as *Markov property* (considering temporal neighborhood, while the Markov properties discussed in Sec. 2.1 considered neighborhood induced by the graph topology). If for all points in time $k \geq 0$ the $p_{ij}^{(k)}$ have the same value p_{ij} (i.e., the transition probabilities do not change over time), the chain is called *homogeneous* and the matrix $\mathbf{P} = (p_{ij})_{i,j\in\Omega}$ is called the *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 $\mu^{(0)}(i) = \Pr(X^{(0)} = i)$, 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 $\boldsymbol{\pi}^{\mathrm{T}} = \boldsymbol{\pi}^{\mathrm{T}}\mathbf{P}$ is called a *stationary distribution*. If the Markov chain at time k has reached the stationary distribution $\boldsymbol{\mu}^{(k)} = \boldsymbol{\pi}$, then 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 $\forall i, j \in \Omega$

$$\pi(i)p_{ij} = \pi(j)p_{ji} \ . \quad (12)$$

This is called the *detailed balance condition*.

Especially relevant are Markov chains for which there exists a unique stationary distribution. For a finite state space 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 $\Pr(X^{(k)} = j \,|\, X^{(0)} = i) > 0$.

A chain is called *aperiodic* if every state can reoccur at irregular times. Formally, a chain is aperiodic if for all $i \in \Omega$ the greatest common divisor of all elements in the set $\{k \in \mathbb{N}_0 \,|\, \Pr(X^{(k)} = i \,|\, X^{(0)} = i) > 0\}$ is 1. One can show that an irreducible and aperiodic Markov chain on a finite state space is guaranteed to converge to its stationary distribution. Let for two distributions α and β on a finite state space Ω the *distance of variation* be defined as

$$d_V(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}|\boldsymbol{\alpha} - \boldsymbol{\beta}| = \frac{1}{2}\sum_{x\in\Omega} |\alpha(x) - \beta(x)| \ . \quad (13)$$

To ease the notation, we allow both row and column probability vectors as arguments of the functions in (13). Then we have:

Theorem 2. *Let π be the stationary distribution of an irreducible and aperiodic Markov chain on a finite state space with transition matrix $\mathbf{P}$. For an arbitrary starting distribution μ,*

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

For a proof see, for instance, [7].

Markov chain Monte Carlo methods make use of this convergence theorem for producing samples from a 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 k is large enough, the state $x^{(k)}$ of $X^{(k)}$ from the constructed chain is then approximately a sample from π and therefore from q. Gibbs sampling [18] is such a MCMC method and will be described in the following section.

3.2 Gibbs sampling

Gibbs sampling is a simple MCMC algorithm for producing samples from the joint probability distribution of multiple random variables. The basic idea is to construct a Markov chain by updating each variable based on its conditional distribution given the state of the others. In the following, we will describe this procedure by explaining how Gibbs sampling can be used to produce samples (approximately) from the Gibbs distribution of an MRF.

We consider an MRF $\boldsymbol{X} = (X_1, \dots, X_N)$ w.r.t. an undirected graph $G = (V, E)$, where $V = \{1, \dots, N\}$ 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 over time, we can consider $X = \{\boldsymbol{X}^{(k)} \,|\, k \in \mathbb{N}_0\}$ as a Markov chain taking values in $\Omega = \Lambda^N$. Then $\boldsymbol{X}^{(k)} = (X_1^{(k)}, \dots, X_N^{(k)})$ describes the state of the MRF at time $k \geq 0$. Between two successive points in time, the new state of the chain is produced by the following procedure. First, a variable X_i, $i \in V$ is randomly picked with a probability $q(i)$ given by a strictly positive probability distribution q on V. Then, the new state for X_i is sampled based on its conditional probability distribution given the state $(x_v)_{v \in V \setminus i}$ of all other variables $(X_v)_{v \in V \setminus i}$. We have $\pi\left(\boldsymbol{x}_i \,|\, (x_v)_{v \in V \setminus i}\right) = \pi\left(\boldsymbol{x}_i \,|\, (x_w)_{w \in \mathcal{N}_i}\right)$ because of the local Markov property of MRFs (cf. Sec. 2.1). The transition probability p_{xy} for two states $\boldsymbol{x}, \boldsymbol{y}$ of the MRF $\boldsymbol{X}$ with $\boldsymbol{x} \neq \boldsymbol{y}$ is

$$p_{\boldsymbol{xy}} = \begin{cases} q(i)\pi\left(y_i \,|\, (x_v)_{v \in V \setminus i}\right), & \text{if } \exists i \in V \text{ so that } \forall v \in V \text{ with } v \neq i\text{: } x_v = y_v \\ 0, & \text{else .} \end{cases} \tag{15}$$

And the probability, that the state of the MRF $\boldsymbol{x}$ stays the same, is $p_{\boldsymbol{xx}} = \sum_{i \in V} q(i)\pi\left(x_i \,|\, (x_v)_{v \in V \setminus i}\right)$.

Convergence of the Gibbs chain. To show that the Markov chain defined by these transition probabilities (the so called Gibbs chain) converges to the joint distribution π of the MRF, we have to prove that π is the stationary distribution of the Gibbs chain and that the chain is irreducible and aperiodic (see Theorem 2).

It is easy to see that π is the stationary distribution by showing that the detailed balance condition (12) 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, then this follows from the fact that $p_{\boldsymbol{yx}} = p_{\boldsymbol{xy}} = 0$. Assume now 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

$$\begin{aligned} \pi(\boldsymbol{x})p_{\boldsymbol{xy}} = \pi(\boldsymbol{x})q(i)\pi\left(y_i \,|\, (x_v)_{v \in V \setminus i}\right) &= \pi\left(x_i, (x_v)_{v \in V \setminus i}\right) q(i) \frac{\pi\left(y_i, (x_v)_{v \in V \setminus i}\right)}{\pi\left((x_v)_{v \in V \setminus i}\right)} \\ &= \pi\left(y_i, (x_v)_{v \in V \setminus i}\right) q(i) \frac{\pi\left(x_i, (x_v)_{v \in V \setminus i}\right)}{\pi\left((x_v)_{v \in V \setminus i}\right)} = \pi(\boldsymbol{y})q(i)\pi\left(x_i \,|\, (x_v)_{v \in V \setminus i}\right) = \pi(\boldsymbol{y})p_{\boldsymbol{yx}} \ . \end{aligned} \tag{16}$$

Thus, the detailed balance condition is fulfilled and π is the stationary distribution.

Since π is strictly positive, so are the conditional probability distributions of the single variables. This means 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, so 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 guarantee 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 in a fixed predefined order. The corresponding algorithm is often referred to as the *periodic Gibbs sampler*. If $\mathbf{P}$ 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 [7]):

$$|\boldsymbol{\mu}\mathbf{P}^k - \boldsymbol{\pi}| \leq \frac{1}{2}|\boldsymbol{\mu} - \boldsymbol{\pi}|(1 - e^{-N\triangle})^k, \tag{17}$$

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 (13).

Gibbs sampling and Metropolis-Hastings algorithms. Gibbs sampling belongs to the broader class of Metropolis-Hastings algorithms [19], see [42] for a good overview. All MCMC algorithms of this class generate the transitions of a Markov chain in two substeps. In the first substep, a candidate state is picked at random from a so called *proposal* distribution. In the second substep, the candidate state is accepted as the new state of the Markov chain with an *acceptance probability* ensuring that detailed balance holds. The proposal distribution of Gibbs sampling always suggests to flip the current state of a single random variable and accepts this with the conditional probability of the suggested state given the states of the remaining random variables.

For sampling in Ising models, the same proposal distribution ("flip the state") has been combined with the acceptance probability $\min(1, \frac{\pi(\boldsymbol{x}')}{\pi(\boldsymbol{x})})$, where $\boldsymbol{x}$ denotes the current and $\boldsymbol{x}'$ the new state of the Markov chain. As discussed in [42], this sampling algorithm may be advantageous over Gibbs sampling. Recently, it has been shown that this indeed also holds true for RBMs [8].

4 Restricted Boltzmann machines

An RBM (also denoted as a Harmonium [49]) is an MRF associated with a bipartite undirected graph as shown in Fig. 5. It consists of m visible units $\boldsymbol{V} = (V_1, \ldots, V_m)$ representing the observable data, and n hidden units $\boldsymbol{H} = (H_1, \ldots, H_n)$ to capture the dependencies between the 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

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \ . \tag{18}$$

For all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, w_{ij} is a real valued weight associated with the edge between the 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.

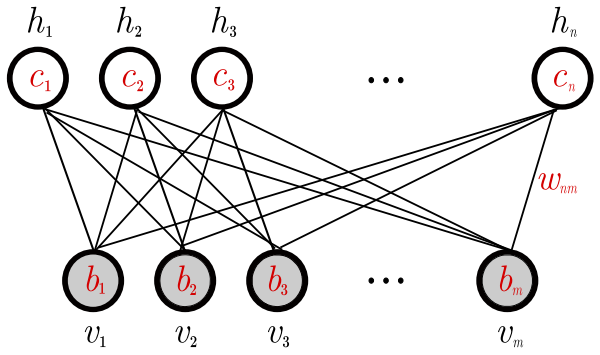

Fig. 5. The network graph of an RBM with n hidden and m visible units.

The graph of an RBM has connections only between the layer of hidden and the layer of 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:

$$p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \prod_{i=1}^{n} p(h_i \,|\, \boldsymbol{v}) \;\text{ and }\; p(\boldsymbol{v} \,|\, \boldsymbol{h}) = \prod_{j=1}^{m} p(v_j \,|\, \boldsymbol{h}) \;\; . \tag{19}$$

Thus, due to the absence of connections between hidden variables, the conditional distributions $p(\boldsymbol{h} \,|\, \boldsymbol{v})$ and $p(\boldsymbol{v} \,|\, \boldsymbol{h})$ factorize nicely, and simple expressions for the factors will be given in Sec. 4.1.

The conditional independence between the variables in the same 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 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*.

Now, how does the RBM distribution over $\boldsymbol{V}$ (e.g., the space of images) look like? The marginal distribution (7) of the visible variables becomes

$$\begin{aligned} p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} = \frac{1}{Z} \sum_{h_1} \sum_{h_2} \cdots \sum_{h_n} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} e^{h_i \left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)} \\ = \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \sum_{h_1} e^{h_1 \left(c_1 + \sum_{j=1}^{m} w_{1j} v_j\right)} \sum_{h_2} e^{h_2 \left(c_2 + \sum_{j=1}^{m} w_{2j} v_j\right)} \cdots \sum_{h_n} e^{h_n \left(c_n + \sum_{j=1}^{m} w_{nj} v_j\right)} \\ = \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} \sum_{h_i} e^{h_i \left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)} = \frac{1}{Z} \prod_{j=1}^{m} e^{b_j v_j} \prod_{i=1}^{n} \left(1 + e^{c_i + \sum_{j=1}^{m} w_{ij} v_j}\right) \;\; . \end{aligned} \tag{20}$$

This equation shows why a (marginalized) RBM can be regarded as a *product of experts* model [21, 58], in which a number of "experts" for the 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 [33]. It has been shown recently that even fewer units can be sufficient, depending on the patterns in the support set [41].

4.1 RBMs and neural networks

The RBM can be interpreted as a stochastic neural network, where the 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 $\mathrm{sig}(x) = 1/(1+e^{-x})$, because

$$p(H_i = 1 \,|\, \boldsymbol{v}) = \mathrm{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right) \tag{21}$$

and

$$p(V_j = 1 \,|\, \boldsymbol{h}) = \mathrm{sig}\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right) \;\; . \tag{22}$$

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

$$\alpha_l(\boldsymbol{h}) = -\sum_{i=1}^{n} w_{il} h_i - b_l \tag{23}$$

and

$$\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) = -\sum_{i=1}^{n} \sum_{j=1, j\neq l}^{m} w_{ij} h_i v_j - \sum_{j=1, j\neq l}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \ . \tag{24}$$

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]:

$$\begin{aligned}
p(V_l = 1 \,|\boldsymbol{h}) &= p(V_l = 1 \,|\boldsymbol{v}_{-l}, \boldsymbol{h}) = \frac{p(V_l = 1, \boldsymbol{v}_{-l}, \boldsymbol{h})}{p(\boldsymbol{v}_{-l}, \boldsymbol{h})} \\
&= \frac{e^{-E(v_l=1, \boldsymbol{v}_{-l}, \boldsymbol{h})}}{e^{-E(v_l=1, \boldsymbol{v}_{-l}, \boldsymbol{h})} + e^{-E(v_l=0, \boldsymbol{v}_{-l}, \boldsymbol{h})}} = \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha_l(\boldsymbol{h})} + e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 0 \cdot \alpha_l(\boldsymbol{h})}} \\
&= \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})} + e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})}} = \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot \left(e^{-\alpha_l(\boldsymbol{h})} + 1\right)} \\
&= \frac{e^{-\alpha_l(\boldsymbol{h})}}{e^{-\alpha_l(\boldsymbol{h})} + 1} = \frac{\frac{1}{e^{\alpha_l(\boldsymbol{h})}}}{\frac{1}{e^{\alpha_l(\boldsymbol{h})}} + 1} = \frac{1}{1 + e^{\alpha_l(\boldsymbol{h})}} = \mathrm{sig}(-\alpha_l(\boldsymbol{h})) = \mathrm{sig}\left(\sum_{i=1}^{n} w_{il} h_i + b_l\right)
\end{aligned} \tag{25}$$

As mentioned in the introduction, an RBM can be reinterpreted as a standard feed-forward neural network with one layer of nonlinear 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.

4.2 The gradient of the log-likelihood

As shown in Sec. 2.2, 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:

$$\begin{aligned}
\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} &= \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j = \sum_{\boldsymbol{h}} \prod_{k=1}^{n} p(h_k \,|\, \boldsymbol{v}) h_i v_j = \sum_{h_i} \sum_{\boldsymbol{h}_{-i}} p(h_i \,|\, \boldsymbol{v}) p(\boldsymbol{h}_{-i} \,|\, \boldsymbol{v}) h_i v_j \\
&= \sum_{h_i} p(h_i \,|\, \boldsymbol{v}) h_i v_j \underbrace{\sum_{\boldsymbol{h}_{-i}} p(\boldsymbol{h}_{-i} \,|\, \boldsymbol{v})}_{=1} = p(H_i = 1 \,|\, \boldsymbol{v}) v_j = \mathrm{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right) v_j
\end{aligned} \tag{26}$$

Since the second term in (9) (i.e., the expectation of the energy gradient under the RBM distribution) 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 (26) the derivative of the log-likelihood of a single training pattern $\boldsymbol{v}$ w.r.t. the weight w_{ij} becomes

$$\begin{aligned}
\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial w_{ij}} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v}, \boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} \\
&= \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j = p(H_i = 1 | \, \boldsymbol{v}) v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_i = 1 | \, \boldsymbol{v}) v_j \ .
\end{aligned} \tag{27}$$

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

$$\begin{aligned}\frac{1}{\ell}\sum_{\boldsymbol{v}\in S}\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial w_{ij}} &= \frac{1}{\ell}\sum_{\boldsymbol{v}\in S}\left[-\mathbb{E}_{p(\boldsymbol{h}\,|\,\boldsymbol{v})}\left[\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}\right] + \mathbb{E}_{p(\boldsymbol{h},\boldsymbol{v})}\left[\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}\right]\right] \\ &= \frac{1}{\ell}\sum_{\boldsymbol{v}\in S}\left[\mathbb{E}_{p(\boldsymbol{h}\,|\,\boldsymbol{v})}\left[v_i h_j\right] - \mathbb{E}_{p(\boldsymbol{h},\boldsymbol{v})}\left[v_i h_j\right]\right] \\ &= \langle v_i h_j\rangle_{p(\boldsymbol{h}\,|\,\boldsymbol{v})q(\boldsymbol{v})} - \langle v_i h_j\rangle_{p(\boldsymbol{h},\boldsymbol{v})} \end{aligned} \tag{28}$$

with q denoting the empirical (or data) distribution. This gives the often stated rule:

$$\sum_{\boldsymbol{v}\in S}\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial w_{ij}} \propto \langle v_i h_j\rangle_{\text{data}} - \langle v_i h_j\rangle_{\text{model}} \tag{29}$$

Analogously to (27) we get the derivatives w.r.t. the bias parameter b_j of the jth visible variable

$$\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial b_j} = v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) v_j \tag{30}$$

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

$$\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial c_i} = p(H_i = 1|\,\boldsymbol{v}) - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_i = 1|\,\boldsymbol{v}) \ . \tag{31}$$

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 (27), (30), and (31)—one can approximate this expectation by samples from the model distribution. These samples can, for example, 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 the log-likelihood gradient using MCMC methods typically requires many sampling steps. However, it has been shown that estimates obtained after running the chain for just a few steps can be sufficient for model training [21]. This leads to *contrastive divergence* (CD) learning, which has become a standard way to train RBMs [21, 4, 24, 3, 23].

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 running 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 subsequently sampling $\boldsymbol{v}^{(t+1)}$ from $p(\boldsymbol{v} \,|\, \boldsymbol{h}^{(t)})$. The gradient, see (9), w.r.t. $\boldsymbol{\theta}$ of the log-likelihood for one training pattern $\boldsymbol{v}^{(0)}$ is then approximated by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad . \tag{32}$$

The derivatives in the direction of each single parameter are obtained by "estimating" the expectations over $p(\boldsymbol{v})$ in (27), (30), and (31) by the single sample $\boldsymbol{v}^{(k)}$. A batch version of CD-k can be seen in Algorithm 1. In *batch learning*, the complete training data set S is used to compute or approximate the gradient in every step. However, it can be more efficient to consider only a subset $S' \subset S$ in every iteration, which reduces the computational burden between parameter updates. The subset S' is called a *mini-batch.* If in every step only a single element of the training set is used to estimate the gradient, the process is often referred to as *online learning.*

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)
    for j = 1,...,m do
        Δb_j ← Δb_j + v_j^(0) − v_j^(k)
    for i = 1,...,n do
        Δc_i ← Δc_i + p(H_i = 1 | v^(0)) − p(H_i = 1 | v^(k))
```

Usually the stationary distribution is not reached after k sampling steps. Thus, $\boldsymbol{v}^{(k)}$ is not a sample from the model distribution and therefore the approximation (32) is biased. Obviously, the bias vanishes as $k \to \infty$.

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 kth 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

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p(\boldsymbol{v}^{(0)}) = &-\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \\ &+ E_{p(\boldsymbol{v}^{(k)} \,|\, \boldsymbol{v}^{(0)})} \left[\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \right] + E_{p(\boldsymbol{v}^{(k)} \,|\, \boldsymbol{v}^{(0)})} \left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial \boldsymbol{\theta}} \right] \end{aligned} \tag{33}$$

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

The first two terms in equation (33) just correspond to the expectation of the CD approximation (under p_k) and the bias is given by the final term.

The approximation error not only depends on the number k of sampling steps, 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 and is determined by the transition probabilities of the chain. The mixing rate of the Gibbs chain of an RBM depends on the magnitude of the model parameters [21, 9, 3, 14]. This becomes clear by considering that the transition probabilities, that is, the conditional probabilities $p(v_j \mid \boldsymbol{h})$ and $p(h_i \mid \boldsymbol{v})$, are given by thresholding $\sum_{i=1}^{n} w_{ij} h_i + b_j$ and $\sum_{j=1}^{m} w_{ij} v_j + c_i$ by the sigmoid function. If the absolute values of the parameters are high, the conditional probabilities can get close to one or zero. If this happens, the states of the Gibbs chain get more and more "predictable", and thus the 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 [14]:

Theorem 2 (Fischer and Igel [14]) *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*

$$\left| E_{q(\boldsymbol{v}^{(0)})} \left[E_{p(\boldsymbol{v}^{(k)} \mid \boldsymbol{v}^{(0)})} \left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] \right] \right| \leq \frac{1}{2} |q - p| \left(1 - e^{-(m+n)\Delta} \right)^k \tag{34}$$

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 [62] 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 [37]. The empirical comparisons of the CD approximation and the true gradient for RBMs (small enough that the gradient is still tractable) conducted in [9] and [3] show 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 the experiments in Sec. 7) in the sense that the model systematically gets worse if k is not large [13]. This is especially bad because the log-likelihood is not tractable in reasonably 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, the weight decay has no effect. If it is too large, the learning converges to models with low likelihood [13] (see Fig. 8 in Sec. 7).

More recently proposed learning algorithms try to obtain better approximations of the log-likelihood gradient by sampling from a Markov chain with a greater mixing rate.

5.2 Persistent contrastive divergence

The idea of *persistent contrastive divergence* (PCD) [55] is described in [61] for log-likelihood maximization of general MRFs and is applied to RBMs in [55]. The PCD approximation is obtained from the CD approximation (32) by replacing the sample $v^{(k)}$ by a sample from a Gibbs chain that is independent of 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 [61, 55]. 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 form, 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) [56]. Fast PCD tries to reach a 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 the *fast* parameters. This 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 $\tilde{p}(H_i = 1 \,|\, \boldsymbol{v}) = \operatorname{sig}\left(\sum_{j=1}^{m}(w_{ij} + w_{ij}^f)v_j + (c_i + c_i^f)\right)$ and $\tilde{p}(V_j = 1 \,|\, \boldsymbol{h}) = \operatorname{sig}\left(\sum_{i=1}^{n}(w_{ij} + w_{ij}^f)h_i + (b_j + b_j^f)\right)$ instead of the conditional probabilities given by (21) and (22). The learning update rule for the fast parameters is the same as 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 has been suggested that regularizing just the fast weights is sufficient [56].

Neither PCD nor FPCD seem to increase 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 [13].

5.3 Parallel tempering

One of the most promising sampling techniques used for RBM training so far is *parallel tempering* (PT) [44, 12, 10]. It introduces supplementary Gibbs chains that sample from more and more smoothed replicas of the original distribution. This can be formalized in the following way. Given an ordered set of M temperatures $1 = T_1 < T_2 < \cdots < T_M$, we define a set of M Markov chains with stationary distributions

$$p_r(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z_r} e^{-\frac{1}{T_r} E(\boldsymbol{v}, \boldsymbol{h})} \tag{35}$$

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. With increasing temperature the probability mass of the Gibbs distribution (35) gets more and more equally distributed (or smoother), and thus the mixing of the corresponding Markov chain gets more and more facilitated. As $T \to \infty$, the uniform distribution is reached, in which subsequent samples of the Gibbs chain are independent from each other and thus the stationary distribution is reached after just one sampling step.

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 [36] */`
  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 (37)
  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 (37)
  for $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**
    $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 \,|\, \boldsymbol{v}) \cdot v_j - p(H_i = 1 \,|\, \boldsymbol{v}_1^{(k)}) \cdot v_{1,j}^{(k)}$
  for $j = 1, \ldots, m$ **do**
    $\Delta b_j \leftarrow \Delta b_j + v_j - v_{1,j}^{(k)}$
  for $i = 1, \ldots, n$ **do**
    $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 \,|\, \boldsymbol{v}) - p(H_i = 1 \,|\, \boldsymbol{v}_1^{(k)})$

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,

$$\min\left\{1, \frac{p_r\big(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1}\big) p_{r-1}\big(\boldsymbol{v}_r, \boldsymbol{h}_r\big)}{p_r\big(\boldsymbol{v}_r, \boldsymbol{h}_r\big) p_{r-1}\big(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1}\big)}\right\} \ , \tag{36}$$

which gives, for RBMs,

$$\min\left\{1, \exp\left(\left(\frac{1}{T_r} - \frac{1}{T_{r-1}}\right) \cdot \left(E(\boldsymbol{v}_r, \boldsymbol{h}_r) - E(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1})\right)\right)\right\} \ . \tag{37}$$

After performing these swaps between chains, which increase the mixing rate, we take the (eventually exchanged) sample $\boldsymbol{v}_1$ of the original chain (with temperature $T_1 = 1$) as a sample from the RBM distribution. This procedure is repeated L times, yielding the 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.

Recently, several approaches to improving PT for RBMs have been suggested. In [11] it is shown how the number M of parallel chains and the values of the temperatures used can be adapted automatically. Other work has been focused on increasing the swapping rate by allowing samples to swap

not only between neighboring chains [6], and on using all samples (not only those of the first chain) to approximate the gradient by weighted averages [6, 15].

Compared to CD, PT introduces computational overhead, but produces a more quickly mixing Markov chain, and thus a less biased gradient approximation. This can lead to better learning, as shown in the experiments in Sec. 7.

6 RBMs with real-valued variables

So far, we have only considered 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 [24], 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.

When keeping the energy function as given in (18) 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 (25). Using the same notation and writing the energy as $E(\boldsymbol{v}, \boldsymbol{h}) = \beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) + v_l \alpha_l(\boldsymbol{h})$, we have

$$\begin{aligned} p(v_l \,|\, \boldsymbol{h}) = p(v_l \,|\, \boldsymbol{v}_{-l}, \boldsymbol{h}) &= \frac{p(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})}{p(\boldsymbol{v}_{-l}, \boldsymbol{h})} \\ &= \frac{e^{-E(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})} I_{[0,1]}}{\int e^{-E(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})} I_{[0,1]}(v_l) \mathrm{d}v_l} = \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l)}{\int e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l) \mathrm{d}v_l} \\ &= \frac{e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l)}{\int e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l) \mathrm{d}v_l} \ , \end{aligned} \tag{38}$$

where the characteristic function $I_{[0,1]}(v_l)$ is 1 if $v_l \in [0, 1]$ and 0 otherwise. That (38) is a truncated exponential with parameter $\alpha_l(\boldsymbol{h})$ can be seen from dropping the restriction to the interval $[0, 1]$, which yields $\frac{e^{-v_l \alpha_l(\boldsymbol{h})}}{\int_0^\infty e^{-v_l \alpha_l(\boldsymbol{h})} \mathrm{d}v_l} = \alpha_l(\boldsymbol{h}) e^{-v_l \alpha_l(\boldsymbol{h})}$.

Widely used are Gaussian-Binary-RBMs where the visible variables given the state of the binary hidden units are normally distributed. Visible neurons with a Gaussian distributed conditionals are gained (in an analogous way to (38)) by augmenting the energy with quadratic terms, which can be written as

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} h_i \frac{v_j}{\sigma_j^2} + \sum_{j=1}^{m} \frac{(v_j - b_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n} c_i h_i \ , \tag{39}$$

see [10].

Making use of the factorization as in (20), the partition function of the Gaussian-Binary-RBMs can be written as

$$\begin{aligned} Z = \sum_{\boldsymbol{h}} \int e^{\sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} h_i \frac{v_j}{\sigma_j^2} - \sum_{j=1}^{m} \frac{(v_j - b_j)^2}{2\sigma_j^2} + \sum_{i=1}^{n} c_i h_i} \mathrm{d}\boldsymbol{v} \\ = \sum_{\boldsymbol{h}} e^{\sum_{i=1}^{n} c_i h_i} \prod_{j=1}^{m} \int e^{\sum_{i=1}^{n} w_{ij} h_i \frac{v_j}{\sigma_j^2} - \frac{(v_j - b_j)^2}{2\sigma_j^2}} \mathrm{d}v_j \\ = \sum_{\boldsymbol{h}} e^{\sum_{i=1}^{n} c_i h_i} \prod_{j=1}^{m} \sqrt{2\pi\sigma_j^2} e^{\frac{2 b_j \sum_i w_{ij} h_i + (\sum_i w_{ij} h_i)^2}{2\sigma_j^2}} \ . \end{aligned} \tag{40}$$

This yields a tractable expression when the number of hidden variables is small enough (e.g., to visualize the log-likelihood in experiments such as shown in Sec. 7).

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 and binary hidden units is rather constrained [57]. However, if we add an additional layer of binary latent variables, we can model any strictly positive density over a compact domain with arbitrary high accuracy [30].

More generally, it is possible to cover continuous valued variables by extending the definition of an RBM to any MRF whose energy function satisfies $p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \prod_i p(h_i \,|\, \boldsymbol{v})$ 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 [24], 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. [59] 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 Experiments

This section will present experiments illustrating the behavior of RBMs in practice. After an introduction to the experimental setup, we will consider sampling from a trained RBM solving an inpainting task. Then, an experiment will show the difference between CD learning with and without "Rao-Blackwellization" [50]. After that, typical learning curves will be shown for different parameter settings of CD and PT. Finally, we will look at the receptive fields learned by RBM hidden units.

All experiments in this section can be reproduced using the open source machine learning library Shark [27], which implements most of the models and algorithms that were discussed.

7.1 Experimental setup

The experiments were performed on two benchmark problems taken from the literature. As a toy problem we considered a 4×4 variant of the Bars-and-Stripes (BAS) benchmark [38, 26]. Each observation corresponds to a square of 4×4 units. The data set is generated by randomly choosing for each pattern an orientation (vertical or horizontal) with equal probability first, and then picking the state for all units of every row or column uniformly at random. Thus, the data set consists out of 32 patterns, six of which can be seen in the top of Fig. 6. Furthermore, we considered the MNIST handwritten digit recognition benchmark [35]. The training set consists out of 60000 samples of digits, see the bottom of Fig. 6 for examples. Each image consists out of 28×28 grayscale pixels, which are binarized with a threshold value of 127.

For training, the RBMs were initialized with small random weights and zero bias parameters. In the BAS experiments, the number of hidden units was set to 16. If not stated otherwise, 20 hidden units were used for modeling the MINIST data. The models were trained using gradient ascent on CD-k with $k \in \{1, 2, 4, 10, 100\}$ or PT with $M \in \{4, 5, 10, 50\}$. The temperatures used in the parallel chains were chosen such that the inverse temperatures were equally distributed over $[0, 1]$ (which may not be the optimal choice [11]). If not stated otherwise, the update rule used for gradient ascent was equal to the one resulting from (6) by replacing the log-likelihood by either the mean of the CD- or the PT-approximation over the training batch. For BAS, standard batch learning was used, while for MNIST the training data was split into mini-batches of 100 and 600 samples in the experiments in Sec. 7.5 and Sec. 7.4, respectively. The learning rate was $\eta = 0.1$ for BAS when training with CD and 0.05 when training with PT and $\eta = 0.3$ for CD-learning on MNIST. To keep the number of hyperparameters small, we did not use a momentum term ($\nu = 0$). In our experience, using momentum

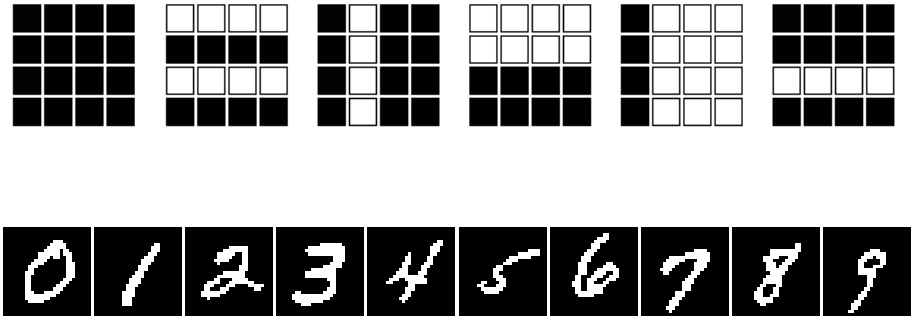

Fig. 6. Top: Patterns from the BAS data set. Bottom: Images from the MNIST data set.

does not qualitatively change the results of the reported experiments. If not stated otherwise, no weight decay term was used ($\lambda = 0$). The experiments in Sec. 7.3 and Sec. 7.4 were repeated 25 times.

7.2 Application example: Reconstruction of images

As outlined in the introduction, a trained RBM can be used as a generative model of the targe distribution. That is, we can use it to sample from $p(\boldsymbol{V})$. If the visible units correspond to pixels from an image, this means that we can generate images from (an approximation of) the target distribution. Now consider the case where we observe only a part of the image, say $V_1 = v_1, \ldots, V_o = v_o$, $o < m$. Then we can use the trained RBM for image inpainting by sampling from $p(V_{o+1}, \ldots, V_m \,|\, V_1 = v_1, \ldots, V_o = v_o)$. To do so, we clamp the observed variables to the observation, that is, we fix $V_1 = v_1, \ldots, V_o = v_o$, and sample the states of the remaining variables, for example using Gibbs sampling.

This is demonstrated in Fig. 7. We trained an RBM on BAS. At different stages of the training procedure, we clamped the first column of the input image to a certain pattern and sampled from the RBM (all other states were initialized with 0.5 and Gibbs sampling was employed). In the beginning, when the RBM distribution was almost uniform, the samples did not resemble the target distribution, as shown in the first row of Fig. 7. After some training, the samples started to reveal the structure of the BAS distribution. As the model distribution got close to the training distribution, the RBM successfully reconstructed the BAS pattern in a few sampling steps, as can be seen in the third row of Fig. 7.

This toy experiment indicates how an RBM can be used for classification as illustrated in Fig. 2. The RBM can be trained on the joint probability distributed of the data and the corresponding labels. After training, a new image is clamped to the corresponding units (i.e., the corresponding visible units are fixed to the pixel values), and the label units are sampled. Another possibility of course is to use the RBM weights to initialize a feed-forward neural network augmented with an output layer corresponding to the labels, which can then be fine tuned in a supervised way for classification.

7.3 To sample or not to sample?

Algorithm 1 differs in a small detail from the description of the CD algorithm in [2]. To approximate the first sum of the log-likelihood gradient we use the probabilities $p(H_i = 1 \,|\, \boldsymbol{v}^{(0)})$, $i = 1, \ldots, n$, exactly

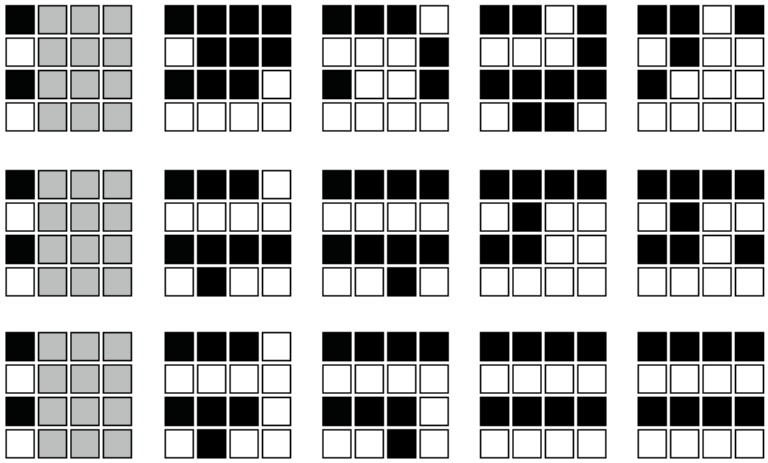

Fig. 7. Reconstruction of an incomplete image by sampling from an RBM. Each row shows the initialization of the visible units and the first four samples from the Gibbs chain. First row: After one iteration of batch learning (log-likelihood $356,154$). Second row: After 1500 iterations of batch learning (log-likelihood $210,882$). Third row: After 4000 iterations of batch learning (log-likelihood $154,618$).

(e.g., see lines 10 and 14 in Algorithm 1), while in [2] this quantity is approximated by the $h_i^{(0)}$ from the Gibbs sampling procedure.

To see the difference, RBMs were trained on BAS using the two different approaches and the log-likelihood was calculated in every iteration. The results are shown in Fig. 8.

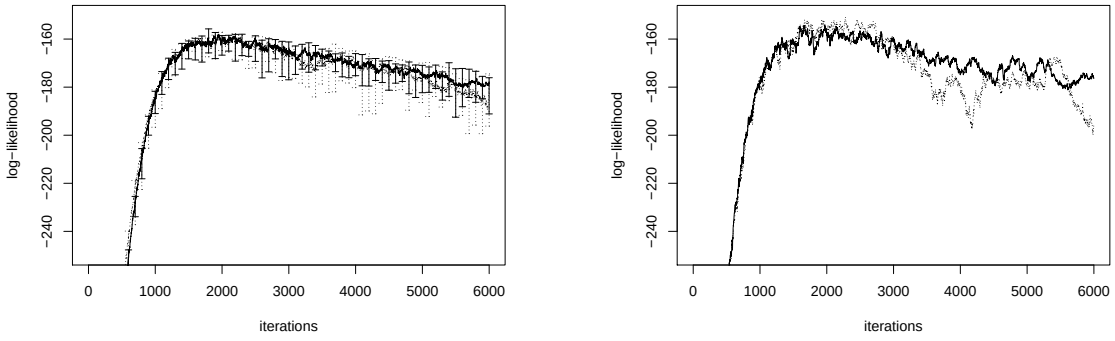

Fig. 8. Evolution of the log-likelihood during training of an RBM with CD-1 where the first expectation is calculated directly (solid line) or approximated by a sample (dashed line). Left: Medians over 25 runs, error bars indicate quartiles. Right: Exemplary single runs.

Both procedures led to similar log-likelihood values, but using the expectation instead of the sample reduced the variance of both the log-likelihood values in a single trial (i.e., oscillations were reduced) as well as in between the different trials as indicated by the error bars in Fig. 8. This result was to be expected, the additional sampling noise increases the variance, while using the expectation reduces the variance of the estimator. The latter is clear from the Rao-Blackwell theorem as also mentioned in the recommended review by Swersky et al. [50].

7.4 Learning curves using CD and PT

The following experiments shall give an idea about the evolution of the log-likelihood during RBM training. The considered RBMs have so few neurons that the log-likelihood is tractable and can be used to show the learning process over time.

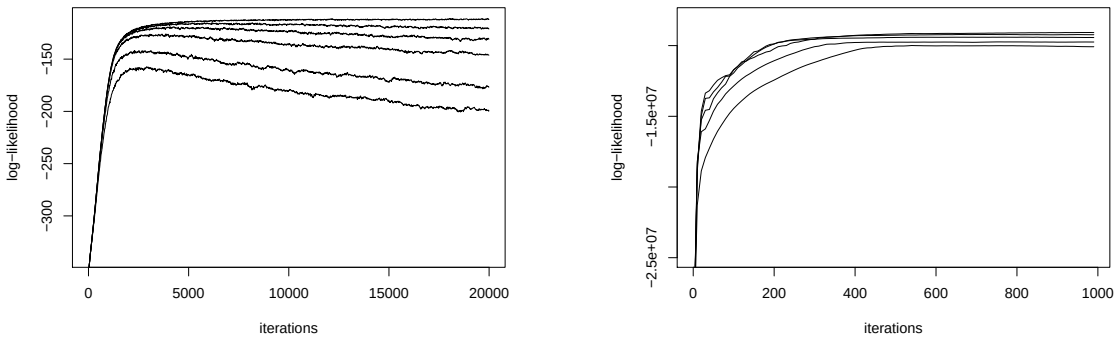

Fig. 9. Evolution of the log-likelihood during training of RBMs with CD-k where different values for k were used. The left plot shows the results for BAS (from bottom to top $k = 1, 2, 5, 10, 20, 100$) and the right plot for MNIST (from bottom to top $k = 1, 2, 5, 10, 20$). The values are medians over 25 runs.

The learning curves of CD-k with different numbers of sampling steps k are depicted in Fig. 9. Shown are the median values over 25 trials. As could be observed in the BAS example, for some learning problems, the log-likelihood steadily decreases after an initial increase if k is not large enough. Thus, after some iterations of the learning process the model starts to get worse. This behavior can be explained by the increasing magnitude of the weights: as explained in Sec. 5.1, the mixing rate of the Gibbs chain decreases with increasing absolute values of the weights and thus the CD approximation gets more and more biased. As suggested by Theorem 1 and Theorem 2, the larger k the less biased the approximation gets. Accordingly, the experiments show that the larger k the less severe the divergence and the higher the maximum log-likelihood value reached during training. The effect of weight-decay with different weight-decay parameters λ is shown in the left plot in Fig. 10. The results indicate that the choice of the λ is crucial. If chosen correctly, the divergence problem was solved. But if the hyperparameter λ was too large, learning stagnated on a low log-likelihood level and thus the RBMs did not model the target distribution accurately. And if the parameter was too small, the weight decay term could not prevent divergence.

As can be seen in the right plot of Fig. 10, the performance of PT on BAS clearly depends on the number of Gibbs chains used in parallel. The more chains are used, the better the mixing, leading to better gradient approximations and thus better learning. Compared to CD-1, PT-1 (i.e., PT with $k = 1$ sampling step performed in every chain at each learning iteration) led to significant higher likelihood

values, but needed more computational power. However, PT-1 with $M = 4$ was comparable to CD with $k = 10$ in terms of model quality, but computationally less demanding. It seems that divergence problems can be prevented with PT if the number of parallel chains is not too small.

Of course, the wallclock runtime of the learning algorithms strongly depends on the implementation. However, to give an idea about actual runtimes we will report some measurements.[3] All results refer to the median of 5 trials. In our experiments, learning BAS with 1000 iterations of CD-1 took 0.14 s. Changing to CD-10 increased the runtime to 0.8 s. Using PT with $M = 5$ instead required 0.67 s for the same number of iterations. Increasing the number of chains to $M = 20$ increased the runtime to 2.18 s. Running CD-1 on MNIST for 100 iterations over mini-batches with 600 samples took 84.67 s for RBMs having 500 hidden units.

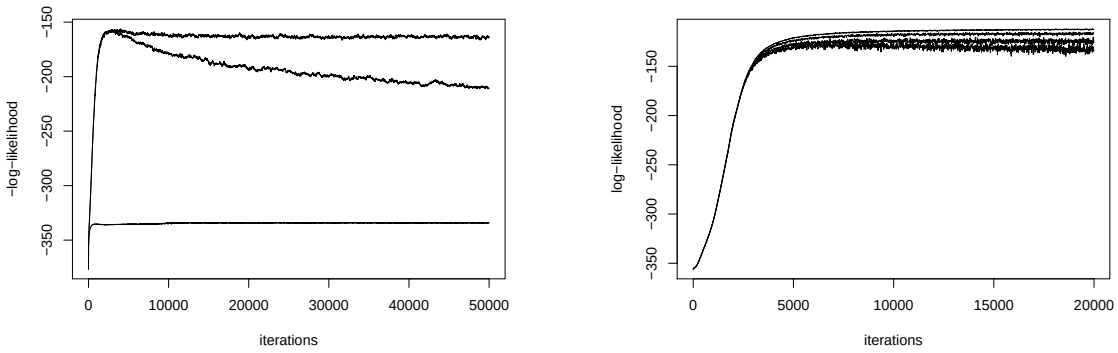

Fig. 10. Evolution of the log-likelihood during training of an RBM on BAS. Left plot: with CD-1 and different values of the weight decay parameter (from bottom to top: $\lambda = 0.05, 0, 00005, 0, 0005$). Right Plot: with PT with different numbers M of temperatures (from bottom to top $M = 4, 5, 10, 50$). The values correspond to the medians over 25 runs.

7.5 Hidden units as filters

As mentioned in the introduction, the hidden units can be viewed as feature detectors. Let the visible units correspond to the pixels of an image. Then we can ask how observed images should look like in order to activate a certain hidden unit most strongly (i.e., to lead to a high probability of this unit being 1). To answer this question, we color-code the weights of the hidden unit and plot them arranged as the corresponding visible units. The resulting image visualizes the preferred input pattern, which is referred to as the learned filter or, in biological terms, the receptive field of the hidden neuron.

Figure 11 shows the weights of RBMs with 16 and 100 hidden units trained on MNIST for 100 epochs. Each square corresponds to the 784 weights of one hidden neuron to the $28 \times 28 = 784$ visible neurons. The squares are ordered according to the probabilities of the corresponding hidden units to be 1 given the training set in decreasing order.

When only 16 hidden units were used, the learned filters are rather complex and it is even possible to recognize digits in them. When 100 hidden units were used, the receptive fields get more localized and show stroke-like features.

[3] The experiments were conducted on a computer with an Intel Core i3 CPU with 3.07GHz running Ubuntu 3.2.0. We used the Shark library [27] and gcc 4.8.0, and the learning ran in a single thread.

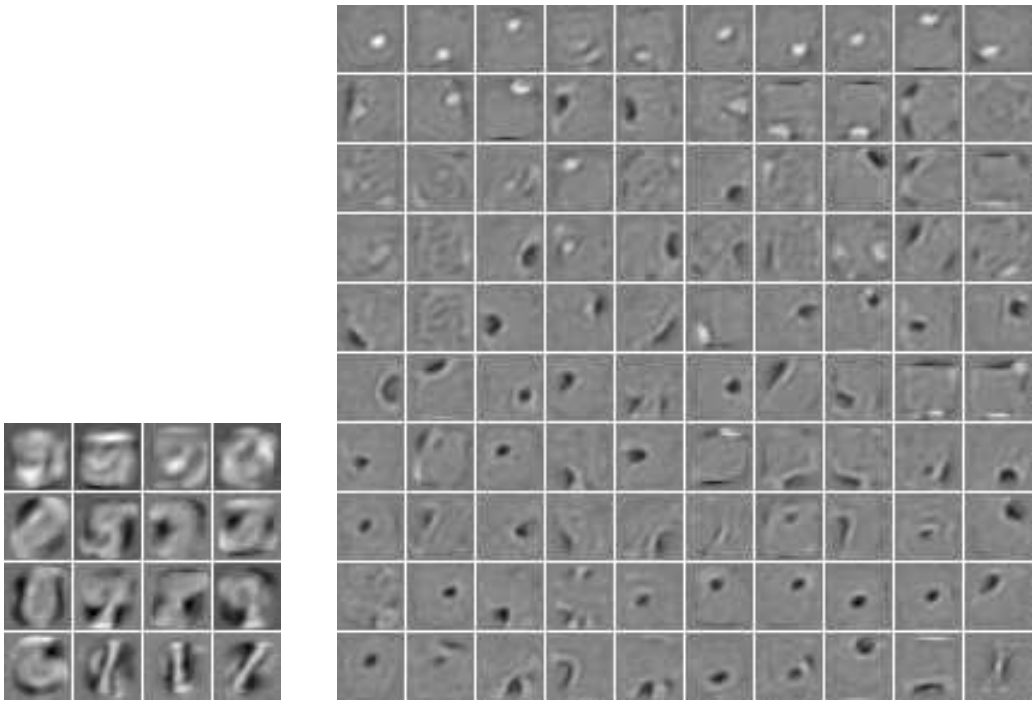

Fig. 11. Visualization of the weights of RBMs with 16 and 100 hidden units (left and right plot, respectively) trained on MNIST. In the two plots, each square image shows the weights of one hidden unit. These images have the size of the input images, and each weight is visualized at the position of the corresponding visible unit. The gray values represent the size of the weights.

8 Where to go from here?

The previous sections have introduced the most basic RBM models. However, several generalizations and extensions of RBMs exist. A notable example are *conditional RBMs* (e.g., [54, 39]). 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. An obvious generalization is to remove the "R" from the RBM, which brings us back to the original Boltzmann machine [1]. The graph of a BM corresponds to the graph of an RBM with additional connections between the variables of one layer. These dependencies make sampling more complicated (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 [45].

The goal of this article was to introduce RBMs from the probabilistic graphical model perspective. It is meant to supplement existing tutorials [2, 50, 20], and it is biased in the sense that it focuses on material that we have 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.

Acknowledgments

This article extends our tutorial in [16]. We thank Jan Melchior for providing Fig. 11. This work has been supported by 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"). Christian Igel gratefully acknowledges support from the European Commission through project AKMI (PCIG10-GA-2011-303655).

References

1. D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.
2. Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 21(6):1601–1621, 2009.
3. Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.
4. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, and U. Montreal. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing (NIPS 19)*, pages 153–160. MIT Press, 2007.
5. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
6. P. Brakel, S. Dieleman, and B. Schrauwen. Training restricted Boltzmann machines with multi-tempering: harnessing parallelization. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 287–292. Evere, Belgium: d-side publications, 2012.
7. P. Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer-Verlag, 1999.
8. K. Brügge, A. Fischer, and C. Igel. The flip-the-state transition operator for restricted Boltzmann machines. *Machine Learning* 13, pp. 53-69, 2013.
9. M. Á. Carreira-Perpiñán and G. E. Hinton. On contrastive divergence learning. In R. G. Cowell and Z. Ghahramani, editors, *10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 59–66. The Society for Artificial Intelligence and Statistics, 2005.
10. K. Cho, T. Raiko, and A. Ilin. Parallel tempering is efficient for learning restricted Boltzmann machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 3246–3253. IEEE Press, 2010.
11. G. Desjardins, A. Courville, and Y. Bengio. Adaptive parallel tempering for stochastic maximum likelihood learning of RBMs. In H. Lee, M. Ranzato, Y. Bengio, G. E. Hinton, Y. LeCun, and A. Y. Ng, editors, *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.
12. G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Dellaleau. Parallel tempering for training of restricted Boltzmann machines. *JMLR W&CP: AISTATS 2010*, 9:145–152, 2010.
13. A. Fischer and C. Igel. Empirical analysis of the divergence of Gibbs sampling based learning algorithms for Restricted Boltzmann Machines. In K. Diamantaras, W. Duch, and L. S. Iliadis, editors, *International Conference on Artificial Neural Networks (ICANN)*, volume 6354 of *LNCS*, pages 208–217. Springer-Verlag, 2010.
14. A. Fischer and C. Igel. Bounding the bias of contrastive divergence learning. *Neural Computation*, 23:664–673, 2011.
15. A. Fischer and C. Igel. Parallel tempering, importance sampling, and restricted Boltzmann machines. In *5th Workshop on Theory of Randomized Search Heuristics (ThRaSH 2011)*, 2011. Online abstract.
16. A. Fischer and C. Igel. An introduction to restricted Boltzmann machines. In L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, editors, *Proceedings of the 17th Iberoamerican Congress on Pattern Recognition (CIARP)*, volume 7441 of *LNCS*, pages 14–36. Springer, 2012.
17. P. V. Gehler, A. D. Holub, and M. Welling. The rate adapting poisson model for information retrieval and object recognition. In W. Cohen and A. Moore, editors, *Proceedings of 23rd International Conference on Machine Learning (ICML)*, pages 337–344. ACM, 2006.
18. D. Geman, S.and Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.
19. W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
20. G. Hinton. A practical guide to training restricted Boltzmann machines. Technical Report UTML TR 2010003, Department of Computer Science, University of Toronto, 2010.
21. G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.
22. G. E. Hinton. Boltzmann machine. *Scholarpedia*, 2(5):1668, 2007.
23. G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434, 2007.
24. G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

25. G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
26. G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, pages 282–317. MIT Press, 1986.
27. C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.
28. J. Kivinen and C. Williams. Multiple texture Boltzmann machines. *JMLR W&CP: AISTATS 2012*, 22:638–646, 2012.
29. D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
30. O. Krause, A. Fischer, T. Glasmachers, and C. Igel. Approximation properties of DBNs with binary hidden units and real-valued visible units. *JMLR W&CP: ICML 2013*, 28(1):419426, 2013.
31. H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *International Conference on Machine learning (ICML)*, pages 536–543. ACM, 2008.
32. S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.
33. N. Le Roux and Y. Bengio. Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.
34. N. Le Roux, N. Heess, J. Shotton, and J. M. Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.
35. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
36. N. M. Lingenheil, R. Denschlag, G. Mathias, and P. Tavan. Efficiency of exchange schemes in replica exchange. *Chemical Physics Letters*, 478:80 84, 2009.
37. D. J. C. MacKay. Failures of the one-step learning algorithm. Cavendish Laboratory, Madingley Road, Cambridge CB3 0HE, UK. `http://www.cs.toronto.edu/~mackay/gbm.pdf`, 2001.
38. D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.
39. V. Mnih, H. Larochelle, and G. E. Hinton. Conditional restricted Boltzmann machines for structured output prediction. In F. G. Cozman and A. Pfeffer, editors, *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, page 514. AUAI Press, 2011.
40. A. Mohamed and G. E. Hinton. Phone recognition using restricted Boltzmann machines. In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 4354–4357. IEEE Press, 2010.
41. G. Montufar and N. Ay. Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines. *Neural Computation*, 23(5):1306–1319, 2011.
42. R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.
43. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, pages 318–362. MIT Press, 1986.
44. R. Salakhutdinov. Learning in Markov random fields using tempered transitions. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS 22)*, pages 1598–1606, 2009.
45. R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. *JMLR W&CP: AISTATS 2009*, 5:448–455, 2009.
46. R. Salakhutdinov and G. E. Hinton. Replicated softmax: an undirected topic model. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS 22)*, pages 1607–1614, 2009.
47. R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 791–798. ACM, 2007.
48. T. Schmah, G. E. Hinton, R. S. Zemel, S. L. Small, and S. C. Strother. Generative versus discriminative training of RBMs for classification of fMRI images. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS 21)*, pages 1409–1416, 2009.
49. P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, pages 194–281. MIT Press, 1986.

50. K. Swersky, B. Chen, B. Marlin, and N. de Freitas. A tutorial on stochastic approximation algorithms for training restricted Boltzmann machines and deep belief nets. In *Information Theory and Applications Workshop (ITA), 2010*, pages 1–10. IEEE, 2010.
51. Y. Tang, R. Salakhutdinov, and G. E. Hinton. Robust Boltzmann machines for recognition and denoising. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2264–2271. IEEE, 2012.
52. G. W. Taylor, E. H. G, and S. Roweis. Modeling human motion using binary latent variables. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19)*, pages 1345–1352. MIT Press, 2007.
53. G. W. Taylor and G. E. Hinton. Factored conditional restricted Boltzmann Machines for modeling motion style. In A. P. Danyluk, L. Bottou, and M. L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 1025–1032. ACM, 2009.
54. G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19)*, pages 1345–1352. MIT Press, 2007.
55. T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *International Conference on Machine learning (ICML)*, pages 1064–1071. ACM, 2008.
56. T. Tieleman and G. E. Hinton. Using fast weights to improve persistent contrastive divergence. In A. Pohoreckyj Danyluk, L. Bottou, and M. L. Littman, editors, *International Conference on Machine Learning (ICML)*, pages 1033–1040. ACM, 2009.
57. N. Wang, J. Melchior, and L. Wiskott. An analysis of Gaussian-binary restricted Boltzmann machines for natural images. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 287–292. Evere, Belgium: d-side publications, 2012.
58. M. Welling. Product of experts. *Scholarpedia*, 2(10):3879, 2007.
59. M. Welling, M. Rosen-Zvi, and G. Hinton. Exponential family harmoniums with an application to information retrieval. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS 17)*, pages 1481–1488. MIT Press, 2005.
60. E. P. Xing, R. Yan, and A. G. Hauptmann. Mining associated text and images with dual-wing harmoniums. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2005.
61. L. Younes. Maximum likelihood estimation of Gibbs fields. In A. Possolo, editor, *Proceedings of an AMS-IMS-SIAM Joint Conference on Spacial Statistics and Imaging*, Lecture Notes Monograph Series. Institute of Mathematical Statistics, Hayward, California, 1991.
62. A. L. Yuille. The convergence of contrastive divergence. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Processing Systems (NIPS 17)*, pages 1593–1600. MIT Press, 2005.