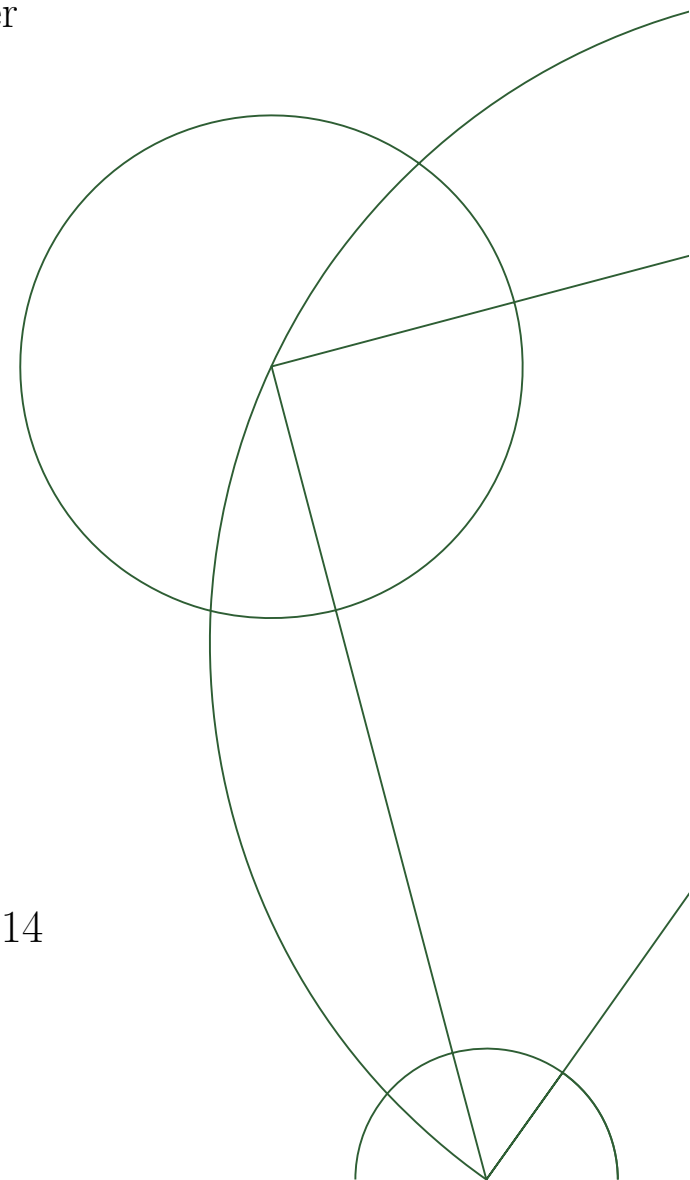
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Training restricted Boltzmann machines

Asja Fischer



June 30th, 2014

This thesis has been submitted to the Department of Computer Science, University of Copenhagen, Denmark in partial fulfillment of the requirements for the degree of Doctor of Philosophy, Ph.D., in computer science. The thesis has been supervised by Christian Igel.

Summary

This thesis is concerned with analyzing and improving training of Restricted Boltzmann Machines (RBMs). It starts with an introduction to RBMs and their most common training methods along with the required concepts from undirected graphical models and Markov chain theory. The second part of the thesis investigates properties of popular training methods in two empirical and two theoretical studies:

- Common learning algorithms for RBMs, like k-step Contrastive Divergence (CD) and its refined variants Persistent CD (PCD) and Fast PCD, rely on gradient ascent on Gibbs sampling based stochastic approximations of the log-likelihood gradient. The approximations are biased, and the bias can lead to a steady decrease of the log-likelihood during learning. In this work these divergence effects are investigated in a detailed empirical study. This includes an analysis of the dependency of the divergence on the number k of Gibbs sampling steps used to gain a sample, the number of hidden variables of the RBM, and on the usage of weight-decay or an adaptive learning rate.

- It was previously reported that despite of the bias the signs of most components of the CD update are equal to the corresponding signs of the log-likelihood gradient. Therefore, training based on resilient backpropagation as an optimization technique depending only on the signs is investigated. However, it does not prevent the divergence caused by the approximation bias.

- The bias of CD depends on k and the mixing rate of the Gibbs chain, which is slowing down with increasing absolute values of the RBM parameters. In this study a new upper bound for the bias is derived that reflects these dependencies and is further affected by the distance in variation between the modeled distribution and the starting distribution of the Gibbs chain.

- One of the most promising sampling techniques used for RBM training so far is Parallel Tempering (PT), which maintains several Gibbs chains in parallel and is designed to produce a faster mixing Markov chain. In this study the convergence rate of PT for sampling from binary RBMs is analyzed by deriving a lower bound on the spectral gap, which shows an exponential dependency on the size of the smallest layer and the sum of the absolute values of the RBM parameters.

The third part of the thesis consists out of three contributions improving different aspects of RBM training:

- A Metropolis-type transition operator is proposed that maximizes the probability of state changes and can replace Gibbs sampling in RBM learning algorithms without

producing computational overhead. It is shown analytically that the operator induces an irreducible, aperiodic Markov chain, and empirically that it leads to faster mixing and in turn to more accurate learning.

- Furthermore, an analysis of centered binary RBMs is given, where centering corresponds to subtracting offset values from visible and hidden variables. It is shown analytically that centering can be reformulated as a different update rule for training normal binary RBMs. The corresponding update direction becomes equivalent to the enhanced gradient for a certain choice of offsets and is invariant to inversions of the data set (generated by flipping each bit) for a broad set of offset values. Numerical simulations show that centering leads to better models in terms of the log-likelihood, and to an update direction that is closer to the natural gradient. Optimal model performance is achieved when subtracting mean values from both visible and hidden variables. It is further shown that the enhanced gradient suffers from divergence more often than other centering variants, which can be prevented by using an exponentially moving average for the offset estimation.

- Assessing model performance is difficult since the likelihood of RBMs is not tractable due to a normalization constant which depends exponentially on the size of the RBM. It can be reliably estimated using Annealed Importance Sampling (AIS), which however needs too much computation time to efficiently monitor the training process. Therefore, alternative techniques from statistical physics for estimating the normalization constant are explored in this study, including Bennett's Acceptance Ratio (BAR). An unifying framework for deriving these methods as well as AIS is given. An empirical analysis shows that BAR gives superior results and outperforms AIS, especially when only a small number of bridging chains are employed.

In the last part of the thesis the representational power of Deep Belief Networks (DBNs) with real valued visible variables is analyzed:

- Deep belief networks are build by stacking RBMs, and known to be able to approximate any distribution over fixed-length binary vectors. However, DBNs are often used for modeling distributions of real valued variables. Therefore, the approximation properties of DBNs with two layers of binary hidden units, and visible units with conditional distributions from the exponential family are analyzed. It is shown that they can, under mild assumptions, model any additive mixture of distributions from the exponential family with independent variables. An arbitrarily good approximation in terms of Kullback-Leibler divergence of an m dimensional mixture distribution with n components can be achieved by a DBN with a layer of m visible variables and n and $n + 1$ hidden variables in the first and second hidden layer, respectively.

Acknowledgements

I want to thank all people who supported me during my work on this thesis. Especially I want to thank my supervisor Christian Igel for his continuous support and advice. Thanks to my colleagues Kai Brügge, Oswin Krause, Jan Melchior, Tobias Glasmachers and Laurenz Wiskott. It was a pleasure and a lot of fun to work with you!

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").

Contents

Acknowledgements **5**

1 Introduction **9**
1.1 What RBMs are good for . . . 9
1.2 Challenges and open questions in RBM training . . . 11
1.3 Outline of the thesis . . . 14

2 Training RBMs: An introduction **17**
2.1 Introduction . . . 18
2.2 Graphical models . . . 21
2.3 Markov chains and Markov chain Monte Carlo techniques . . . 26
2.4 Restricted Boltzmann machines . . . 30
2.5 Approximating the RBM log-likelihood gradient . . . 34
2.6 RBMs with real-valued variables . . . 41
2.7 Experiments . . . 43
2.8 Where to go from here? . . . 49

3 Empirical analysis of the divergence of Gibbs sampling based learning algorithms **51**
3.1 Introduction . . . 52
3.2 Training RBMs . . . 52
3.3 Experiments . . . 55
3.4 Discussion . . . 58
3.5 Conclusion . . . 60

4 Training RBMs based on the signs of the CD approximation **63**
4.1 Introduction . . . 64
4.2 RBMs and CD learning . . . 65
4.3 Training RBMs with resilient backpropagation . . . 65
4.4 Experiments . . . 66

4.5 Results . . . 66
4.6 Discussion and conclusion . . . 68

5 Bounding the bias of contrastive divergence learning 71
5.1 Training RBMs using contrastive divergence . . . 72
5.2 Bounding the CD approximation error . . . 73
5.3 Experimental results . . . 77
5.4 Discussion and conclusion . . . 77

6 A bound for the convergence rate of parallel tempering for sampling RBMs 81
6.1 Introduction . . . 82
6.2 Background . . . 82
6.3 Main result . . . 86
6.4 Bounding the spectral gap of PT . . . 87
6.5 Proof of the main result . . . 89
6.6 Conclusion . . . 97
6.7 Appendix . . . 98

7 The flip-the-state transition operator for RBMs 99
7.1 Introduction . . . 100
7.2 Background . . . 101
7.3 The flip-the-state transition operator . . . 102
7.4 Experiments . . . 108
7.5 Results and discussion . . . 111
7.6 Conclusion . . . 116
7.7 Appendix . . . 117

8 How to center binary RBMs 119
8.1 Introduction . . . 120
8.2 Restricted Boltzmann machines . . . 121
8.3 Centered restricted Boltzmann machines . . . 125
8.4 Initialization of the model parameters . . . 128
8.5 Methods . . . 130
8.6 Results . . . 132
8.7 Conclusion . . . 146
8.8 Appendix . . . 148

9 On Bennett's acceptance ratio for estimating the partition function of RBMs 155
9.1 Introduction . . . 156
9.2 Restricted Boltzmann machines and parallel tempering . . . 157
9.3 Optimal estimators of the normalisation constant for a given sampler . . 158
9.4 Experiments and results . . . 161
9.5 Conclusion . . . 167
9.6 Appendix . . . 168

10 Properties of DBNs with binary hidden and real-valued visible units 171
10.1 Introduction . . . 172
10.2 Background . . . 173
10.3 Approximation properties . . . 175
10.4 Conclusions . . . 183

11 Discussion and conclusion 185
11.1 Summary and discussion . . . 185
11.2 Conclusion . . . 193
11.3 Future work . . . 194

Bibliography 196

List of Publications 209

Chapter 1

Introduction

The field of machine learning is concerned with the development and analysis of algorithms that can learn from data. Learning in this context corresponds to extracting and modeling certain principles underlying the data. Thereby different types of learning problems can be distinguished. Classification and regression tasks are typical supervised learning problems. In a supervised learning problem the training data is given in form of pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots$, where x_i is an input and y_i the corresponding output value (or label). The learning task is now to extract and model the relation between input and output values, that is, to infer a function mapping points in the input space to points in the output space. This function, also referred to as hypothesis, can then be used to find the outputs corresponding to formerly unseen input values. In unsupervised learning the training data consists of unlabeled input points $x_1, x_2, x_3 \ldots$, and the learning problem corresponds to either finding some structure in the data; to generating a representation that summarizes and explains key features of the data; or to building a (generative) model of the data. A resulting representation or model can for example be used for data compression or for prediction and decision making. Typical unsupervised learning tasks include clustering, dimensionality reduction and density estimation.

1.1 What RBMs are good for

This thesis focuses on a particular machine learning model, namely Restricted Boltzmann Machines (RBMs, Smolensky, 1986), which are introduced in detail in Chapter 2 together with their statistical background. Restricted Boltzmann machines are undirected graphical models that can also be interpreted as two-layered stochastic neural networks. As an undirected graphical model, an RBM represents a probability distribu-

tion that can be used in an unsupervised learning problem to model some distribution over some input space. Given a set of samples $x_1, x_2, x_3 \ldots$ as training data, learning corresponds to adjusting the model parameters of the RBM such that the represented probability distribution fits the training data as well as possible. The RBM then forms a model of the distribution underlying the training data.

The trained RBM can be used in various ways, which shall be described in the following.

RBMs as generative models. When an RBM is used as a generative model, it is used for drawing samples from the learned distribution. If the training data consists of images, this can for example be used to generate textures from these images (Le Roux et al., 2011; Courville et al., 2011; Kivinen and Williams, 2012), or to solve inpainting tasks (Kivinen and Williams, 2012; Tang et al., 2012) by sampling the missing or deteriorated parts of a given image from the distribution. Another example is the modeling and generation of human motion patterns (Taylor et al., 2007; Taylor and Hinton, 2009; Sukhbaatar et al., 2011).

RBMs as classifiers. Restricted Boltzmann machines can also be used as classifiers. If labeled training data is given, and the RBM is trained on the joint distribution of inputs and labels, one can sample the missing label for a represented image from the distribution or assign a new image to the class with the highest probability under the model (Salakhutdinov et al., 2007; Larochelle and Bengio, 2008). Furthermore, other techniques for using RBMs as classifiers exist, where several RBMs are trained, each modeling the input data from one class (Schmah et al., 2009). Possible applications are for example the classification of fMRI images (Schmah et al., 2009) or character recognition and text classification problems (Larochelle et al., 2012).

RBMs as feature extractors. Another branch of applications employs RBMs as feature extractors. For feature extraction, one makes use of the fact that RBMs comprise two types of variables: a layer of visible variables which correspond to the components of the inputs, and a layer of hidden (or latent) variables which capture dependencies between the visible neurons. After training, the expected states of the hidden variables given an input can be interpreted as the (learned) features extracted from this input pattern. If the number of hidden units is small, this leads to low dimensional representations, for example of semantic documents which can be utilized in document retrieval (Xing et al., 2005; Gehler et al., 2006; Salakhutdinov and Hinton, 2009a). Another interesting application area where RBMs are employed as feature extractors is for example the field music similarity measuring as it is needed for music recommendation, exploration and classification (Schlüter and Osendorfer, 2011; Tran et al., 2014).

RBMs as building blocks of deep architectures. Restricted Boltzmann machines got into the focus of attention after being proposed as the building blocks of multi-layer generative models which are called Deep Belief Networks (DBNs, Hinton and Salakhutdinov, 2006; Hinton, 2007a). The basic idea underlying these deep architectures is that the features extracted by the hidden neurons of an RBM can serve as input for another RBM. By stacking RBMs in this way, one can learn features from features in the hope of getting a hierarchy of more and more abstract representations. The resulting multi-layer architecture can either be used as a deep generative model (Hinton, 2007a; Hinton et al., 2006) or in a discriminative way. For the latter, the network is regarded as a feed forward neural network (which is possible because of the structural equivalence of RBMs and neural networks) and augmented by a final layer of variables that represent the desired outputs. This multi-layer perceptron can then be fine-tuned in a supervised way by backpropagation (Hinton and Salakhutdinov, 2006; Hinton, 2007a; Bengio et al., 2007; Erhan et al., 2010). Deep belief networks were for example applied with great success to audio (Lee et al., 2009b) and image classification (Ranzato et al., 2007; Larochelle et al., 2007).

1.2 Challenges and open questions in RBM training

Compared to 1986 when RBM have been introduced (Smolensky, 1986), RBMs can now be applied to more interesting problems. This is due to the increase in computational power and the development of new learning strategies which started around 2002 (Hinton, 2002). Training RBMs corresponds to adjusting the parameters as to maximize the probability of the training data under the model. This corresponds to maximizing the likelihood of the parameters given the training data. Maximum likelihood learning is in general challenging for undirected probabilistic graphical models because the maximum likelihood parameters cannot be found analytically and the log-likelihood gradient needed for a gradient based optimization is not tractable. It involves averages over a number of terms exponential in the size of the model. Obtaining unbiased estimates of these averages by Markov Chain Monte Carlo (MCMC) methods typically requires many sampling steps and thus is computationally too demanding. Hinton (2002) however showed that the biased estimates obtained after running a Gibbs chain for just a few steps are sufficient for RBM training. He suggested to initialize the Gibbs chain with a sample from the training set and usually only one sampling step is applied. The resulting approximation of the log-likelihood gradient is referred to as k-step Contrastive Divergence (CD, or CD-k) and optimization by gradient ascent on the CD approximation still is one of the most popular RBM training techniques (Hinton, 2002).

Against this background my thesis addresses the following challenges and open questions regarding the training of RBMs.

Analyzing biased approximations. Due to the bias of the approximation, CD learning does not necessarily reach a maximum likelihood estimate of the parameters (Carreira-Perpiñán and Hinton, 2005; Bengio and Delalleau, 2009). Yuille (2005) specified conditions under which CD learning is guaranteed to converge to an optimal solution. These conditions however 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 by MacKay (2001). Furthermore, Sutskever and Tieleman (2010) showed that the CD-1 update is not the gradient of any function, and while a regularized CD update has a fixed point for a large class of regularization functions, it is possible to design regularization functions that cause CD learning to cycle indefinitely.

The bias of CD depends not only on the number k of sampling steps, but also on the mixing rate of the Gibbs chain (i.e. the speed of the convergence of the Markov chain to the model distribution) and mixing slows down with increasing magnitude of model parameters (Hinton, 2002; Carreira-Perpiñán and Hinton, 2005; Bengio and Delalleau, 2009). The magnitude of the RBM parameters increases during training, and so does the CD-bias. This can lead to a distortion of the learning process: after some learning iterations, the likelihood can start to diverge in the sense that the model systematically gets worse (Fischer and Igel, 2009; Desjardins et al., 2010b).

A number of refined learning algorithms for RBM training were introduced which make use of different sampling techniques and aim at reducing the bias of the gradient approximation. Persistent CD (PCD, Tieleman, 2008) and Fast PCD (FPCD, Tieleman and Hinton, 2009) are also based on Gibbs sampling, but do not reinitialize the chain with a training sample in each iteration. Training RBMs based on the sampling technique Parallel Tempering (PT), also known as replica exchange or tempered MCMC sampling, seems especially promising (Salakhutdinov, 2009; Desjardins et al., 2010b; Cho et al., 2010). PT is designed to overcome the limitations of Metropolis-Hasting algorithms (like Gibbs sampling) when sampling from multi-modal target distributions and as a result to lead to faster mixing Markov chains (Swendsen and Wang, 1986; Geyer, 1991). Consistently, empirical studies show that using PT for RBM training results in better generative models and can prevent the likelihood from diverging (Desjardins et al., 2010b; Cho et al., 2010). But a theoretical analysis of PT based training is missing so far.

Analyzing and increasing the mixing rate of sampling methods. Since the bias of the gradient approximations heavily depends on the mixing rate of the Markov

chain employed for drawing samples, it is of high interest to use sampling techniques with a fast convergence rate in RBM training algorithms. Likewise, when using RBMs as generative models, one is interested in sampling techniques leading to a fast convergence of the Markov chain to the stationary distribution, since this is the distribution one wishes to draw samples from.

Therefore, both an analysis of the mixing rate of sampling methods applied to RBMs as well as the development of faster mixing sampling techniques are required. While a well known upper bound for the convergence rate of Gibbs sampling as employed by CD and (F)PCD exists (see, e.g. Brémaud (1999)) and can easily be applied to RBMs, the convergence rate of PT applied to RBMs still needs to be investigated.

Making the learning process more robust against changes of the data representation. An undesired property of training RBMs based on the log-likelihood gradient is that the learning procedure is not invariant to the data representation. For example training an RBM on the MNIST data set of handwritten digits (white digits on black background) leads to a better model than training it on the data set generated by flipping each bit (black digits on white background). So far two ways of becoming invariant to such changes of the data representation have been described. On the one hand, Cho et al. (2011) designed an alternative update direction referred to as enhanced gradient that is invariant to flips of the data and can replace the gradient in the learning procedure. On the other hand, Tang and Sutskever (2011) have shown empirically that subtracting the data mean from the visible variables leads to similar learning results on flipped and unflipped data sets. Removing the mean of all variables is generally known as the "centering trick", which was originally proposed for feed forward neural networks (LeCun et al., 1998b). Recently, it has also been applied to the visible and hidden variables of Deep Boltzmann Machines (DBMs) where it has been shown to lead to better conditioned optimization problems and to improve some aspects of model performance (Montavon and Müller, 2012).

Assessing model quality. Assessing model quality during RBM training is difficult because the likelihood (like its gradient) is not tractable for large models. Salakhutdinov and Murray (2008) showed that Annealed Importance Sampling (AIS, Neal, 2001) can be used to estimate the likelihood. It is however computationally too demanding to efficiently monitor the training progress and the resulting estimate is not always reliable (Schulz et al., 2010). More recently, Desjardins et al. (2011) introduced an estimation procedure that reuses samples generated by PT during training and can track the likelihood during training without a lot of computational overhead.

Analyzing the representational power of RBMs and DBNs. It is important to ask what kind of distributions a probabilistic model is able to learn. An RBM having binary hidden variables and visible variables with a Gaussian conditional distribution is known to represent a mixture of Gaussian distributions, where the single components are placed on the vertices of a projected parallelotope (Wang et al., 2012). Because of this restriction, the model's capabilities are rather limited. However, RBMs with binary hidden and visible variables are known to be able to approximate any distribution over the input variables arbitrarily well, provided that they have sufficiently many hidden variables (Le Roux and Bengio, 2008; Montufar and Ay, 2011). Furthermore, it was shown that a binary DBN never needs more variables than a binary RBM to model a distribution with a certain accuracy (Le Roux and Bengio, 2010). However, DBNs are frequently applied to model real-valued data, and little is known about their representational power in this case.

1.3 Outline of the thesis

The aim of the work presented in this thesis is to contribute to a deeper understanding of RBMs and their training algorithms and to improve the learning process by addressing the open questions and challenges outlined above. In a bird-eyes view, the thesis starts with an introduction to RBMs and their training algorithms in Chapter 2. The work presented in the Chapters 3 to 6 analyzes properties of existing RBM learning algorithms empirically and theoretically. Chapters 7 and 8 introduce two ways of improving training. Chapter 9 analyzes techniques to estimate the RBM log-likelihood and Chapter 10 investigates approximation properties of DBNs. A final conclusion and discussion is given in Chapter 11.

In more detail and seen in the context of previous work, the thesis can be outlined as follows. It starts with a review article that introduces RBMs and their training algorithms along with the needed concepts from graph and Markov chain theory in Chapter 2. Chapter 3 analyzes the impact of the biased approximations of the log-likelihood gradient used in the CD algorithm and its refined variants PCD and FPCD on the learning process. In this analysis the divergence behavior, formerly observed by Fischer and Igel (2009) and Desjardins et al. (2010b), and its dependencies on different training settings are further investigated. It was reported by Bengio and Delalleau (2009) that despite the bias, the signs of most components of the CD update are equal to the corresponding signs of the log-likelihood gradient. This suggests replacing gradient ascent by an optimization technique such as resilient backpropagation (Riedmiller, 1994; Igel and Hüsken, 2003), which only depends on the signs. This idea is investigated in Chapter 4. The bias of CD is theoretically analyzed in Chapter 5

by deriving a new upper bound on the expectation of the CD approximation error under the empirical distribution. Chapter 6 presents the first theoretical analysis of the convergence rate of the advanced sampling technique PT applied to binary RBMs. Based on general results on the mixing rate of PT (Woodard et al., 2009b), a lower bound for the spectral gap of PT for sampling in RBMs is derived, giving rise to an upper bound of the convergence rate. The work presented in Chapter 7 proposes a Metropolis-type transition operator that maximizes the probability of state changes in the hope of producing a faster mixing Markov chain. It can replace Gibbs sampling in RBM learning algorithms and is related to the Metropolized Gibbs sampler often used to sample from Ising models (Neal, 1993; Liu, 1996). Chapter 8 analyzes centered RBMs, where centering corresponds to subtracting offset values from the variables. This work is related to the analysis of centered DBMs by Montavon and Müller (2012) and includes a unifying view on centering and the approaches of Cho et al. (2011) and Tang and Sutskever (2011) to yield a training procedure which is invariant to changes of the data representation. In Chapter 9 different techniques for estimating the likelihood of RBMs are explored. This includes different variants of AIS and Bennett's Acceptance Ratio (BAR) method (Bennett, 1976), which is one of the components of the estimator suggested by Desjardins et al. (2011). The representational power of DBNs with real-valued visible variables is investigated in Chapter 10, which gives an analysis of the approximation properties of DBNs with two layers of binary hidden units and visible units with conditional distributions from the exponential family. Finally, a summarizing discussion and a conclusion are given in Chapter 11.

Organization. This is a cumulative thesis consisting of papers published in or submitted to peer-reviewed journals and conferences. Each publication is assigned a separate chapter, and the content of the publications remained largely unchanged. Only minimal changes were applied to align the notation and achieve a consistent formatting throughout the thesis. Furthermore, all references were gathered in a joint bibliography in the end. Since all papers deal with the analysis of properties or the development and analysis of learning algorithms of the same class of models, there exists some redundancy in the introduction and background sections of the papers. The reader is kindly asked to excuse this and may just skip the redundant parts. Chapters 3, 4, and 5 are partially based on work done during my master thesis.

Chapter 2

Training RBMs: An introduction

This chapter is based on the manuscript "Training restricted Boltzmann machines: An introduction" by A. Fischer and C. Igel published in *Pattern Recognition* 47, pp. 25-39, 2014.

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.

2.1 Introduction

In the last years, models extending or borrowing concepts from *Restricted Boltzmann Machines* (RBMs, Smolensky, 1986) have enjoyed much popularity for pattern analysis and generation, with applications including image classification, processing, and generation (Hinton and Salakhutdinov, 2006; Tang et al., 2012; Le Roux et al., 2011; Kivinen and Williams, 2012; Schmah et al., 2009; Larochelle and Bengio, 2008); learning movement patterns (Taylor et al., 2007; Taylor and Hinton, 2009); collaborative filtering for movie recommendations (Salakhutdinov et al., 2007); extraction of semantic document representations (Salakhutdinov and Hinton, 2009a; Gehler et al., 2006; Xing et al., 2005); and acoustic modeling (Mohamed and Hinton, 2010). 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 (Ackley et al., 1985; Hinton, 2007b). 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 Figure 2.1 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 (Hinton, 2007b). 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.

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 (Le Roux et al., 2011; Kivinen and Williams, 2012)), in particular from the marginal distributions of interest, see right plot of Figure 2.1. For example, we can fix some visible units corresponding to a partial observation (i.e., we set the

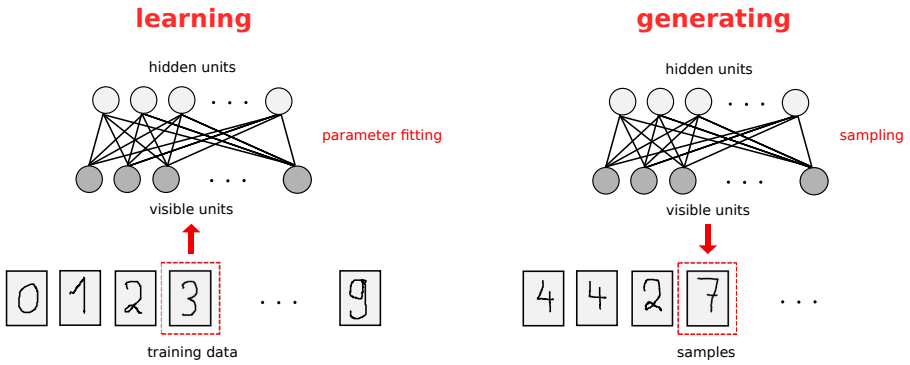


Figure 2.1: Left: Learning an RBM corresponds to fitting its parameters such that the distribution represented by the RBM models the distribution underlying the training data, here handwritten digits. Right: After learning, the trained RBM can be used to generate samples from the learned distribution.

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 (Kivinen and Williams, 2012; Tang et al., 2012), see Figure 7.4 in Section 2.7. 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 Figure 2.2. 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 Figure 2.2.

Compared to the 1980s when RBMs were first introduced (Smolensky, 1986), they can now be applied to more interesting problems due to the increase in computational power and the development of new learning strategies (Hinton, 2002). 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, Hinton and Salakhutdinov, 2006; Hinton, 2007a). 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 Figure 2.3 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

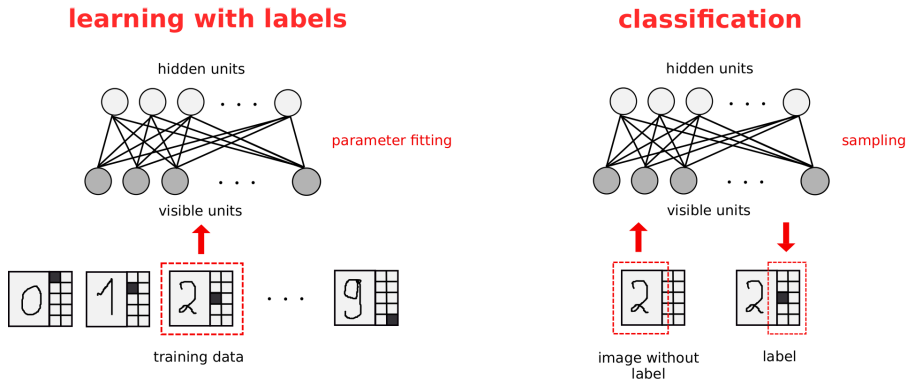


Figure 2.2: Left: RBM trained on labeled data, here images of handwritten digits combined with ten binary indicator variables, one of which is set to 1 indicating that the image shows a particular digit while the others are set to 0. Right: The label corresponding to an input image is obtained by fixing 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.

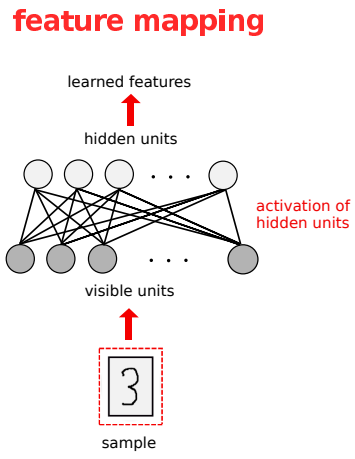


Figure 2.3: The trained RBM can be used as a feature extractor. An input pattern is clamped to the visible neurons. The conditional probabilities of the hidden neurons to be 1 are interpreted as a new representation of the input. This new representation can serve as input to another RBM or to a different learning system.

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 (Rumelhart et al., 1986b). 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 (Hinton and Salakhutdinov, 2006).

Boltzmann machines can be regarded as probabilistic graphical models, namely undirected graphical models also known as Markov Random Fields (MRFs, Koller and Friedman, 2009). The embedding into the framework of probabilistic graphical models provides immediate access to a wealth of theoretical results and well-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 2.2 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 Section 2.3. Then RBMs will be formally described in Section 2.4, and the application of MCMC algorithms to RBMs training will be the topic of Section 2.5. Finally, we will discuss RBMs with real-valued variables before concluding with some experiments.

2.2 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 the book by Bishop (2006). The most comprehensive resource on graphical models is the textbook by Koller and Friedman (2009).

2.2.1 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 Figure 2.4. Here the neighborhood of node v_4 is $\{v_2, v_5, v_6\}$.

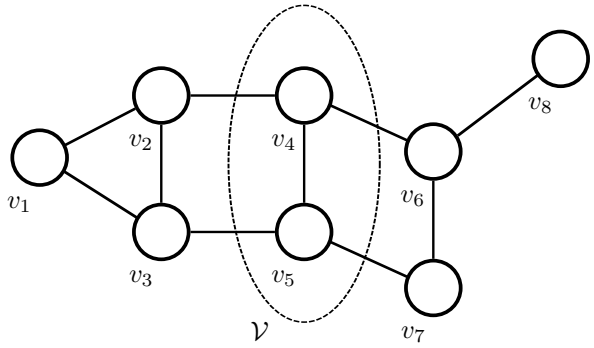


Figure 2.4: An example of an undirected graph. The nodes v_1 and v_8 are separated by $\mathcal{V} = \{v_4, v_5\}$.

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 Figure 2.4, 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 Figure 2.4.

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 the textbooks by Lauritzen (1996) and Koller and Friedman (2009)):

Theorem 2.1. *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

$$\forall \boldsymbol{x}, \hat{\boldsymbol{x}} \in \Lambda^{|V|} : (x_c)_{c \in C} = (\hat{x}_c)_{c \in C} \Rightarrow \psi_C(\boldsymbol{x}) = \psi_C(\hat{\boldsymbol{x}}) \tag{2.1}$$

and

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}). \tag{2.2}$$

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

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C) = \frac{1}{Z} e^{\sum_{C \in \mathcal{C}} \ln \psi_C(\boldsymbol{x}_C)} = \frac{1}{Z} e^{-E(\boldsymbol{x})} \ , \tag{2.3}$$

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

(2.3), which is also called the *Gibbs distribution*.

2.2.2 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

$$\ln \mathcal{L}(\boldsymbol{\theta} \,|\, S) = \ln \prod_{i=1}^{\ell} p(\boldsymbol{x}_i \,|\, \boldsymbol{\theta}) = \sum_{i=1}^{\ell} \ln p(\boldsymbol{x}_i \,|\, \boldsymbol{\theta}) \ . \tag{2.4}$$

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

$$\mathrm{KL}(q||p) = \sum_{\boldsymbol{x} \in \Omega} q(\boldsymbol{x}) \ln \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} = \sum_{\boldsymbol{x} \in \Omega} q(\boldsymbol{x}) \ln q(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in \Omega} q(\boldsymbol{x}) \ln p(\boldsymbol{x}) \ . \tag{2.5}$$

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 (2.5), 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:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \underbrace{\eta \frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} \Big(\ln \mathcal{L}(\boldsymbol{\theta}^{(t)} \mid S) \Big) - \lambda \boldsymbol{\theta}^{(t)} + \nu \Delta \boldsymbol{\theta}^{(t-1)}}_{= \Delta \boldsymbol{\theta}^{(t)}} \tag{2.6}$$

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 (2.6). 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 (Rumelhart et al., 1986b).

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

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \ , \tag{2.7}$$

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 (2.7) with parameters $\boldsymbol{\theta}$, the log-likelihood given a single training example $\boldsymbol{v}$ is

$$\ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v}) = \ln p(\boldsymbol{v} \,|\, \boldsymbol{\theta}) = \ln \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} = \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{2.8}$$

and for the gradient we get

$$\begin{aligned}
\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right) - \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right) \\
&= -\frac{1}{\sum\limits_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \frac{1}{\sum\limits_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} \\
&= -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad .
\end{aligned} \tag{2.9}$$

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

$$p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \frac{p(\boldsymbol{v},\boldsymbol{h})}{p(\boldsymbol{v})} = \frac{\frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\frac{1}{Z} \sum\limits_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\sum\limits_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \quad . \tag{2.10}$$

Note that the last expression of (2.9) 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.

2.3 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 the book by Brémaud (1999) and the aforementioned textbooks by Bishop (2006) and Koller and Friedman (2009). An emphasis will be put on Gibbs sampling as an MCMC technique often used for MRF training and in particular for training RBMs.

2.3.1 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

$$p_{ij}^{(k)} = \Pr\left(X^{(k+1)} = j \,|\, X^{(k)} = i, X^{(k-1)} = i_{k-1}, \ldots, X^{(0)} = i_0\right) = \Pr\left(X^{(k+1)} = j \,|\, X^{(k)} = i\right) \quad . \tag{2.11}$$

The "memorylessness" of a stochastic process expressed by (2.11) is also referred to as *Markov property* (considering temporal neighborhood, while the Markov properties discussed in Section 2.2.1 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 $\forall i, j \in \Omega$

$$\pi(i)p_{ij} = \pi(j)p_{ji} \quad . \tag{2.12}$$

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

$$d_V(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}|\boldsymbol{\alpha} - \boldsymbol{\beta}| = \frac{1}{2}\sum_{x\in\Omega} |\alpha(x) - \beta(x)| \ . \tag{2.13}$$

To ease the notation, we allow both row and column probability vectors as arguments of the functions in (2.13). Then we have:

Theorem 2.2. *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,*

$$\lim_{k\to\infty} d_V(\boldsymbol{\mu}^T\mathbf{P}^k, \boldsymbol{\pi}^T) = 0 \ . \tag{2.14}$$

For a proof see, for instance, the textbook by Brémaud (1999).

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 (Geman, 1984) is such a MCMC method and will be described in the following section.

2.3.2 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 $\boldsymbol{X} = (X_1, \ldots, X_N)$ w.r.t. an undirected graph $G = (V, E)$, where $V = \{1, \ldots, N\}$ for the sake of clearness of notation. The random variables X_i, $i \in V$ take values in a finite set Λ and $\pi(\boldsymbol{x}) = \frac{1}{Z}e^{-\mathcal{E}(\boldsymbol{x})}$ is the joint probability distribution of $\boldsymbol{X}$. Furthermore, if we assume that the MRF changes its state 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 $\boldsymbol{X}^{(k)} = (X_1^{(k)}, \ldots, X_N^{(k)})$ 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. Section 2.2.1). The transition probability p_{xy} for two states $\boldsymbol{x}, \boldsymbol{y}$ of the MRF $\boldsymbol{X}$ with $\boldsymbol{x} \neq \boldsymbol{y}$ is

$$p_{\boldsymbol{xy}} = \begin{cases} q(i)\pi\left(y_i \,|\, (x_v)_{v\in V\setminus i}\right), & \text{if } \exists i \in V \text{ so that } \forall v \in V \text{ with } v \neq i\text{: } x_v = y_v \\ 0, & \text{else .} \end{cases} \tag{2.15}$$

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 2.2).

It is easy to see that π is the stationary distribution by showing that the detailed balance condition (2.12) 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

$$\begin{aligned} \pi(\boldsymbol{x})p_{\boldsymbol{xy}} &= \pi(\boldsymbol{x})q(i)\pi\left(y_i \,|\, (x_v)_{v\in V\setminus i}\right) = \pi\left(x_i, (x_v)_{v\in V\setminus i}\right) q(i) \frac{\pi\left(y_i, (x_v)_{v\in V\setminus i}\right)}{\pi\left((x_v)_{v\in V\setminus i}\right)} \\ &= \pi\left(y_i, (x_v)_{v\in V\setminus i}\right) q(i) \frac{\pi\left(x_i, (x_v)_{v\in V\setminus i}\right)}{\pi\left((x_v)_{v\in V\setminus i}\right)} = \pi(\boldsymbol{y})q(i)\pi\left(x_i \,|\, (x_v)_{v\in V\setminus i}\right) = \pi(\boldsymbol{y})p_{\boldsymbol{yx}} \ . \end{aligned} \tag{2.16}$$

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 Brémaud, 1999):

$$|\boldsymbol{\mu}\mathbf{P}^k - \boldsymbol{\pi}| \leq \frac{1}{2}|\boldsymbol{\mu} - \boldsymbol{\pi}|(1 - e^{-N\triangle})^k, \tag{2.17}$$

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 (2.13).

Gibbs sampling and Metropolis-Hastings algorithms. Gibbs sampling belongs to the broader class of Metropolis-Hastings algorithms (Hastings, 1970), see the review by Neal (1993) 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 by Neal (1993), this sampling algorithm may be advantageous over Gibbs sampling. Recently, it has been shown that this indeed also holds true for RBMs (Brügge et al., 2013).

2.4 Restricted Boltzmann machines

An RBM (also denoted as a Harmonium (Smolensky, 1986)) is an MRF associated with a bipartite undirected graph as shown in Figure 2.5. 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

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \ . \tag{2.18}$$

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.

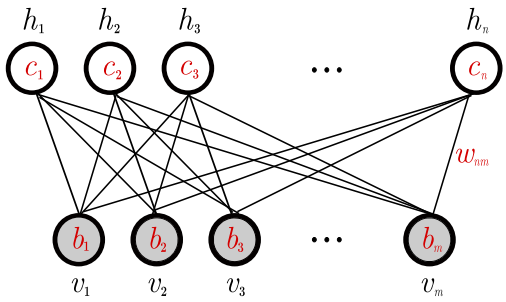


Figure 2.5: The network graph of an RBM with n hidden and m visible units.

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:

$$p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \prod_{i=1}^{n} p(h_i \,|\, \boldsymbol{v}) \;\text{ and }\; p(\boldsymbol{v} \,|\, \boldsymbol{h}) = \prod_{j=1}^{m} p(v_j \,|\, \boldsymbol{h}) \;. \tag{2.19}$$

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 Section 2.4.1.

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 (2.7) of the visible variables becomes

$$\begin{aligned}
p(\boldsymbol{v}) &= \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} = \frac{1}{Z} \sum_{h_1} \sum_{h_2} \cdots \sum_{h_n} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} e^{h_i \left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)} \\
&= \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \sum_{h_1} e^{h_1 \left(c_1 + \sum_{j=1}^{m} w_{1j} v_j\right)} \sum_{h_2} e^{h_2 \left(c_2 + \sum_{j=1}^{m} w_{2j} v_j\right)} \cdots \sum_{h_n} e^{h_n \left(c_n + \sum_{j=1}^{m} w_{nj} v_j\right)} \\
&= \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} \sum_{h_i} e^{h_i \left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)} = \frac{1}{Z} \prod_{j=1}^{m} e^{b_j v_j} \prod_{i=1}^{n} \left(1 + e^{c_i + \sum_{j=1}^{m} w_{ij} v_j}\right) \;.
\end{aligned} \tag{2.20}$$

This equation shows why a (marginalized) RBM can be regarded as a *product of experts* model (Hinton, 2002; Welling, 2007), 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 (Le Roux and Bengio, 2008). It has been shown recently that even fewer units can be sufficient, depending on the patterns in the support set (Montufar and Ay, 2011).

2.4.1 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

$$p(H_i = 1 \,|\, \boldsymbol{v}) = \mathrm{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right) \tag{2.21}$$

and

$$p(V_j = 1 \,|\, \boldsymbol{h}) = \mathrm{sig}\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right) \ . \tag{2.22}$$

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

$$\alpha_l(\boldsymbol{h}) = -\sum_{i=1}^{n} w_{il} h_i - b_l \tag{2.23}$$

and

$$\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) = -\sum_{i=1}^{n} \sum_{j=1, j\neq l}^{m} w_{ij} h_i v_j - \sum_{j=1, j\neq l}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \ . \tag{2.24}$$

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 (Bengio, 2009):

$$\begin{aligned}
p(V_l = 1 \,|\boldsymbol{h}) = p(V_l = 1 \,|\boldsymbol{v}_{-l}, \boldsymbol{h}) &= \frac{p(V_l = 1, \boldsymbol{v}_{-l}, \boldsymbol{h})}{p(\boldsymbol{v}_{-l}, \boldsymbol{h})} \\
= \frac{e^{-E(v_l=1, \boldsymbol{v}_{-l}, \boldsymbol{h})}}{e^{-E(v_l=1, \boldsymbol{v}_{-l}, \boldsymbol{h})} + e^{-E(v_l=0, \boldsymbol{v}_{-l}, \boldsymbol{h})}} &= \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha_l(\boldsymbol{h})} + e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 0 \cdot \alpha_l(\boldsymbol{h})}} \\
= \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})} + e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})}} &= \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-\alpha_l(\boldsymbol{h})}}{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot \left(e^{-\alpha_l(\boldsymbol{h})} + 1\right)} \\
= \frac{e^{-\alpha_l(\boldsymbol{h})}}{e^{-\alpha_l(\boldsymbol{h})} + 1} = \frac{\frac{1}{e^{\alpha_l(\boldsymbol{h})}}}{\frac{1}{e^{\alpha_l(\boldsymbol{h})}} + 1} &= \frac{1}{1 + e^{\alpha_l(\boldsymbol{h})}} = \mathrm{sig}(-\alpha_l(\boldsymbol{h})) = \mathrm{sig}\left(\sum_{i=1}^{n} w_{il} h_i + b_l\right)
\end{aligned} \tag{2.25}$$

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.

2.4.2 The gradient of the log-likelihood

As shown in Section 2.2.2, the log-likelihood gradient of an MRF can be written as the sum of two expectations, see (2.9). For RBMs the first term of (2.9) (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:

$$\begin{aligned}\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} &= \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j = \sum_{\boldsymbol{h}} \prod_{k=1}^{n} p(h_k \,|\, \boldsymbol{v}) h_i v_j \\ &= \sum_{h_i} \sum_{\boldsymbol{h}_{-i}} p(h_i \,|\, \boldsymbol{v}) p(\boldsymbol{h}_{-i} \,|\, \boldsymbol{v}) h_i v_j = \sum_{h_i} p(h_i \,|\, \boldsymbol{v}) h_i v_j \underbrace{\sum_{\boldsymbol{h}_{-i}} p(\boldsymbol{h}_{-i} \,|\, \boldsymbol{v})}_{=1} \\ &= p(H_i = 1 \,|\, \boldsymbol{v}) v_j = \operatorname{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i \right) v_j \quad (2.26)\end{aligned}$$

Since the second term in (2.9) (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 (2.26) the derivative of the log-likelihood of a single training pattern $\boldsymbol{v}$ w.r.t. the weight w_{ij} becomes

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial w_{ij}} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v}, \boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} \\ &= \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j \\ &= p(H_i = 1 | \, \boldsymbol{v}) v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_i = 1 | \, \boldsymbol{v}) v_j \ . \quad (2.27)\end{aligned}$$

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

following notations are used:

$$\begin{aligned}\frac{1}{\ell}\sum_{\boldsymbol{v}\in S}\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial w_{ij}} &= \frac{1}{\ell}\sum_{\boldsymbol{v}\in S}\left[-\mathbb{E}_{p(\boldsymbol{h}\,|\,\boldsymbol{v})}\left[\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}\right]+\mathbb{E}_{p(\boldsymbol{h},\boldsymbol{v})}\left[\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}\right]\right]\\ &= \frac{1}{\ell}\sum_{\boldsymbol{v}\in S}\left[\mathbb{E}_{p(\boldsymbol{h}\,|\,\boldsymbol{v})}\left[v_i h_j\right]-\mathbb{E}_{p(\boldsymbol{h},\boldsymbol{v})}\left[v_i h_j\right]\right]\\ &= \langle v_i h_j\rangle_{p(\boldsymbol{h}\,|\,\boldsymbol{v})q(\boldsymbol{v})}-\langle v_i h_j\rangle_{p(\boldsymbol{h},\boldsymbol{v})}\end{aligned} \tag{2.28}$$

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

$$\sum_{\boldsymbol{v}\in S}\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial w_{ij}} \propto \langle v_i h_j\rangle_{\text{data}}-\langle v_i h_j\rangle_{\text{model}} \tag{2.29}$$

Analogously to (2.27) we get the derivatives w.r.t. the bias parameter b_j of the jth visible variable

$$\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial b_j} = v_j-\sum_{\boldsymbol{v}}p(\boldsymbol{v})v_j \tag{2.30}$$

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

$$\frac{\partial \ln\mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial c_i} = p(H_i=1|\,\boldsymbol{v})-\sum_{\boldsymbol{v}}p(\boldsymbol{v})p(H_i=1|\,\boldsymbol{v})\ . \tag{2.31}$$

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 (2.27), (2.30), and (2.31)—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.

2.5 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.

2.5.1 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 (Hinton, 2002). This leads to *Contrastive Divergence* (CD) learning, which has become a standard way to train RBMs (Hinton, 2002; Bengio et al., 2007; Hinton et al., 2006; Bengio and Delalleau, 2009; Hinton, 2007a).

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 (2.9), w.r.t. $\boldsymbol{\theta}$ of the log-likelihood for one training pattern $\boldsymbol{v}^{(0)}$ is then approximated by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad . \tag{2.32}$$

The derivatives in the direction of each single parameter are obtained by "estimating" the expectations over $p(\boldsymbol{v})$ in (2.27), (2.30), and (2.31) by the single sample $\boldsymbol{v}^{(k)}$.

Algorithm 2.1: 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 $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$
forall the $\boldsymbol{v} \in S$ **do**
  $\boldsymbol{v}^{(0)} \leftarrow \boldsymbol{v}$
  for $t = 0, \ldots, k-1$ **do**
    for $i = 1, \ldots, n$ **do** sample $h_i^{(t)} \sim p(h_i \,|\, \boldsymbol{v}^{(t)})$
    for $j = 1, \ldots, m$ **do** sample $v_j^{(t+1)} \sim p(v_j \,|\, \boldsymbol{h}^{(t)})$
  for $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**
    $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 \,|\, \boldsymbol{v}^{(0)}) \cdot v_j^{(0)} - p(H_i = 1 \,|\, \boldsymbol{v}^{(k)}) \cdot v_j^{(k)}$
  for $j = 1, \ldots, m$ **do**
    $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
  for $i = 1, \ldots, n$ **do**
    $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 \,|\, \boldsymbol{v}^{(0)}) - p(H_i = 1 \,|\, \boldsymbol{v}^{(k)})$

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.*

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 (5.2) is biased. Obviously, the bias vanishes as $k \to \infty$.

The theoretical results from Bengio and Delalleau (2009) 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 2.3 (Bengio and Delalleau, 2009)**.** *For a converging Gibbs chain*

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

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

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}} \ln p(\boldsymbol{v}^{(0)}) = &- \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \\ &+ E_{p(\boldsymbol{v}^{(k)} \,|\, \boldsymbol{v}^{(0)})} \left[\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \right] + E_{p(\boldsymbol{v}^{(k)} \,|\, \boldsymbol{v}^{(0)})} \left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial \boldsymbol{\theta}} \right] \end{aligned} \tag{2.33}$$

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

The first two terms in equation (2.33) 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 (Hinton, 2002; Carreira-Perpiñán and Hinton, 2005; Bengio and Delalleau, 2009; Fischer and Igel, 2011a). This becomes clear by considering that the transition probabilities, that is, the conditional probabilities $p(v_j \,|\, \boldsymbol{h})$ and $p(h_i \,|\, \boldsymbol{v})$, are given by thresholding $\sum_{i=1}^{n} w_{ij} h_i + b_j$ and $\sum_{j=1}^{m} w_{ij} v_j + c_i$ 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 the work of Bengio and Delalleau (2009).

An upper bound on the expectation of the CD approximation error under the empirical distribution is given by the following theorem (Fischer and Igel, 2011a):

Theorem 2.4 (Fischer and Igel, 2011a)**.** *Let p denote the marginal distribution of the visible units of an RBM and let q be the empirical distribution defined by a set of samples $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell$. Then an upper bound on the expectation of the error of the CD-k approximation of the log-likelihood derivative w.r.t some RBM parameter θ_a is given by*

$$\left| E_{q(\boldsymbol{v}^{(0)})} \left[E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] \right] \right| \leq \frac{1}{2} |q - p| \left(1 - e^{-(m+n)\Delta} \right)^k \tag{2.34}$$

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 (2005) 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 by MacKay (2001). The empirical comparisons of the CD approximation and the true gradient for RBMs (small enough that the gradient is still tractable) conducted by Carreira-Perpiñán and Hinton (2005) and Bengio and Delalleau (2009) 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 Section 7.4) in the sense that the model systematically gets worse if k is not large (Fischer and Igel, 2010a). 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 (2.6), 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 (Fischer and Igel, 2010a) (see Figure 2.8 in Section 7.4).

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.

2.5.2 Persistent contrastive divergence

The idea of *Persistent Contrastive Divergence* (PCD, Tieleman, 2008) is described by Younes (1991) for log-likelihood maximization of general MRFs and is applied to RBMs by Tieleman (2008). The PCD approximation is obtained from the CD approximation (5.2) 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 (Younes, 1991; Tieleman, 2008). The number of persistent chains used for sampling (or the number of samples used to approximate the second term of gradient (2.9)) is a hyperparameter 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, Tieleman and Hinton, 2009). 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 (2.21) and (2.22). 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 (Tieleman and Hinton, 2009).

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 by Fischer and Igel (2010a).

2.5.3 Parallel tempering

One of the most promising sampling techniques used for RBM training so far is *Parallel Tempering* (PT, Salakhutdinov, 2009; Desjardins et al., 2010b; Cho et al., 2010). 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

$$p_r(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z_r} e^{-\frac{1}{T_r} E(\boldsymbol{v}, \boldsymbol{h})} \tag{2.35}$$

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 (2.35) 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.

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,

$$\min \left\{ 1, \frac{p_r\big(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1}\big) p_{r-1}\big(\boldsymbol{v}_r, \boldsymbol{h}_r\big)}{p_r\big(\boldsymbol{v}_r, \boldsymbol{h}_r\big) p_{r-1}\big(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1}\big)} \right\} , \tag{2.36}$$

which gives, for RBMs,

$$\min \left\{ 1, \exp\left(\left(\frac{1}{T_r} - \frac{1}{T_{r-1}} \right) \cdot \left(E(\boldsymbol{v}_r, \boldsymbol{h}_r) - E(\boldsymbol{v}_{r-1}, \boldsymbol{h}_{r-1}) \right) \right) \right\} . \tag{2.37}$$

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

Algorithm 2.2: k-step parallel tempering with M temperatures

Input: RBM $(V_1, \dots, V_m, H_1, \dots, H_n)$, training batch S, current state $\boldsymbol{v}_r$ of Markov chain with stationary distribution p_r for $r = 1, \dots, M$

Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \dots, n$, $j = 1, \dots, m$

init $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \dots, n$, $j = 1, \dots, m$
forall the $\boldsymbol{v} \in S$ **do**
  for $r = 1, \dots, M$ **do**
    $\boldsymbol{v}_r^{(0)} \leftarrow \boldsymbol{v}_r$
    for $i = 1, \dots, n$ **do** sample $h_{r,i}^{(0)} \sim p(h_{r,i} \,|\, \boldsymbol{v}_r^{(0)})$
    for $t = 0, \dots, k-1$ **do**
      for $j = 1, \dots, m$ **do** sample $v_{r,j}^{(t+1)} \sim p(v_{r,j} \,|\, \boldsymbol{h}_r^{(t)})$
      for $i = 1, \dots, n$ **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 (Lingenheil et al., 2009) */`
  for $r \in \{s \,|\, 2 \le s \le M \text{ and } s \bmod 2 = 0\}$ **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 (2.37)
  for $r \in \{s \,|\, 3 \le s \le M \text{ and } s \bmod 2 = 1\}$ **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 (2.37)
  for $i = 1, \dots, n$, $j = 1, \dots, m$ **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 $j = 1, \dots, m$ **do**
    $\Delta b_j \leftarrow \Delta b_j + v_j - v_{1,j}^{(k)}$
  for $i = 1, \dots, n$ **do**
    $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 \,|\, \boldsymbol{v}) - p(H_i = 1 \,|\, \boldsymbol{v}_1^{(k)})$

second term in (2.9)). 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. Desjardins et al. (2010a) has 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 (Brakel et al., 2012), and on using all samples (not only those of the first chain) to approximate the gradient by weighted averages (Brakel et al., 2012; Fischer and Igel, 2011b).

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 Section 7.4.

2.6 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 Hinton et al. (2006), 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 (2.18) 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 (2.25). 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

$$\begin{aligned} p(v_l \,|\, \boldsymbol{h}) = p(v_l \,|\, \boldsymbol{v}_{-l}, \boldsymbol{h}) &= \frac{p(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})}{p(\boldsymbol{v}_{-l}, \boldsymbol{h})} \\ &= \frac{e^{-E(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})} I_{[0,1]}}{\int e^{-E(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})} I_{[0,1]}(v_l) \mathrm{d}v_l} = \frac{e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l)}{\int e^{-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})} \cdot e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l) \mathrm{d}v_l} \\ &= \frac{e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l)}{\int e^{-v_l \alpha_l(\boldsymbol{h})} I_{[0,1]}(v_l) \mathrm{d}v_l} \quad , \end{aligned} \tag{2.38}$$

where the characteristic function $I_{[0,1]}(v_l)$ is 1 if $v_l \in [0, 1]$ and 0 otherwise. That (2.38) 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 (2.38)) by augmenting the energy with quadratic terms, which can be written as

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i \frac{v_j}{\sigma_j^2} + \sum_{j=1}^{m} \frac{(v_j - b_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n} c_i h_i \ , \tag{2.39}$$

(see Cho et al., 2010).

Making use of the factorization as in (2.20), the partition function of the Gaussian-Binary-RBMs can be written as

$$\begin{aligned} Z &= \sum_{\boldsymbol{h}} \int e^{\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i \frac{v_j}{\sigma_j^2} - \sum_{j=1}^{m} \frac{(v_j - b_j)^2}{2\sigma_j^2} + \sum_{i=1}^{n} c_i h_i} \, \mathrm{d}\boldsymbol{v} \\ &= \sum_{\boldsymbol{h}} e^{\sum_{i=1}^{n} c_i h_i} \prod_{j=1}^{m} \int e^{\sum_{i=1}^{n} w_{ij} h_i \frac{v_j}{\sigma_j^2} - \frac{(v_j - b_j)^2}{2\sigma_j^2}} \, \mathrm{d}v_j \\ &= \sum_{\boldsymbol{h}} e^{\sum_{i=1}^{n} c_i h_i} \prod_{j=1}^{m} \sqrt{2\pi\sigma_j^2} e^{\frac{2 b_j \sum_i w_{ij} h_i + (\sum_i w_{ij} h_i)^2}{2\sigma_j^2}} \ . \end{aligned} \tag{2.40}$$

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 Section 7.4).

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 (Wang et al., 2012). 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 (Krause et al., 2013).

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 by Hinton et al. (2006), this holds for any energy function of the form

$$E(\boldsymbol{v}, \boldsymbol{h}) = \sum_{i,j} \phi_{i,j}(h_i, v_j) + \sum_{j} \omega_j(v_j) + \sum_{i} \nu_i(h_i) \tag{2.41}$$

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. (2005) 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.

2.7 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" (Swersky et al., 2010). 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 (Igel et al., 2008), which implements most of the models and algorithms that were discussed.

2.7.1 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 (MacKay, 2002; Hinton and Sejnowski, 1986). 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 Figure 2.6. Furthermore, we considered the MNIST handwritten digit recognition benchmark (LeCun et al., 1998a). The training set consists out of 60000 samples of digits, see the bottom of Figure 2.6 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 (Desjardins et al., 2010a)). If not stated otherwise, the update rule used for gradient ascent was equal to the one resulting from (2.6) 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 Section 2.7.5 and Section 2.7.4, 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 ($\nu = 0$).

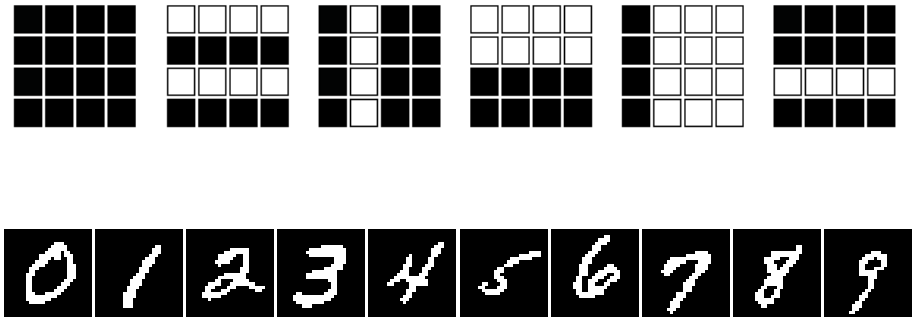


Figure 2.6: Top: Patterns from the BAS data set. Bottom: Images from the MNIST data set.

In our experience, using momentum 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 Section 2.7.3 and Section 2.7.4 were repeated 25 times.

2.7.2 Application example: Reconstruction of images

As outlined in the introduction, a trained RBM can be used as a generative model of the target 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 Figure 2.7. 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 Figure 2.7. 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 Figure 2.7.

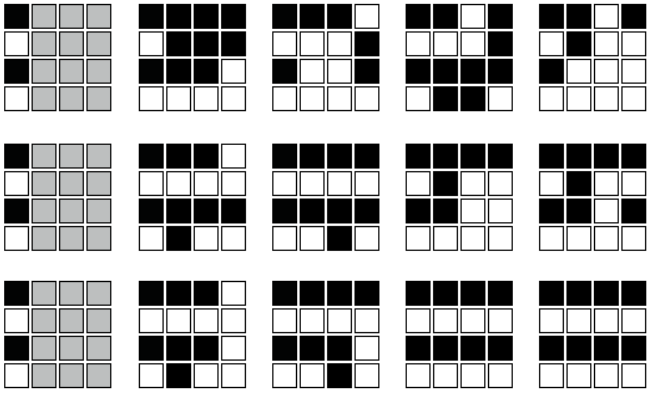


Figure 2.7: Reconstruction of an incomplete image by sampling from an RBM. Each row shows the initialization of the visible units and the first four samples from the Gibbs chain. First row: After one iteration of batch learning (log-likelihood $356,154$). Second row: After 1500 iterations of batch learning (log-likelihood $210,882$). Third row: After 4000 iterations of batch learning (log-likelihood $154,618$).

This toy experiment indicates how an RBM can be used for classification as illustrated in Figure 2.2. 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.

2.7.3 To sample or not to sample?

Algorithm 1 differs in a small detail from the description of the CD algorithm in article by Bengio (2009). 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 (e.g., see lines 8 and 12 in Algorithm 1), while in the article by Bengio (2009) 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 Figure 2.8.

Both procedures led to similar log-likelihood values, but using the expectation

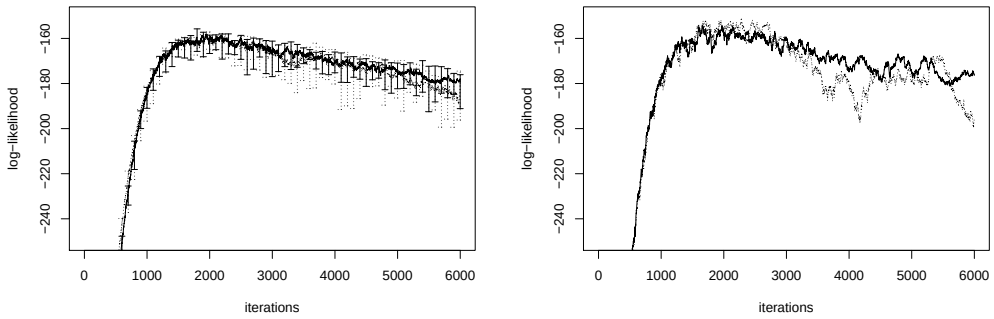


Figure 2.8: Evolution of the log-likelihood during training of an RBM with CD-1 where the first expectation is calculated directly (solid line) or approximated by a sample (dashed line). Left: Medians over 25 runs, error bars indicate quartiles. Right: Exemplary single runs.

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 Figure. 2.8. 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. (2010).

2.7.4 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.

The learning curves of CD-k with different numbers of sampling steps k are depicted in Figure 2.9. 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 Section 2.5.1, 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 2.3 and Theorem 2.4, 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

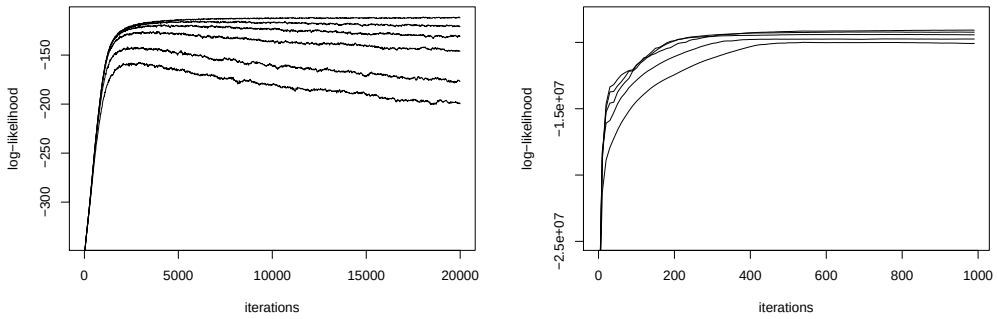


Figure 2.9: Evolution of the log-likelihood during training of RBMs with CD-k where different values for k were used. The left plot shows the results for BAS (from bottom to top $k = 1, 2, 5, 10, 20, 100$) and the right plot for MNIST (from bottom to top $k = 1, 2, 5, 10, 20$). The values are medians over 25 runs.

with different weight-decay parameters λ is shown in the left plot in Figure 2.10. 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 Figure 2.10, 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.[1] 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

[1]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 (Igel et al., 2008) and gcc 4.8.0, and the learning ran in a single thread.

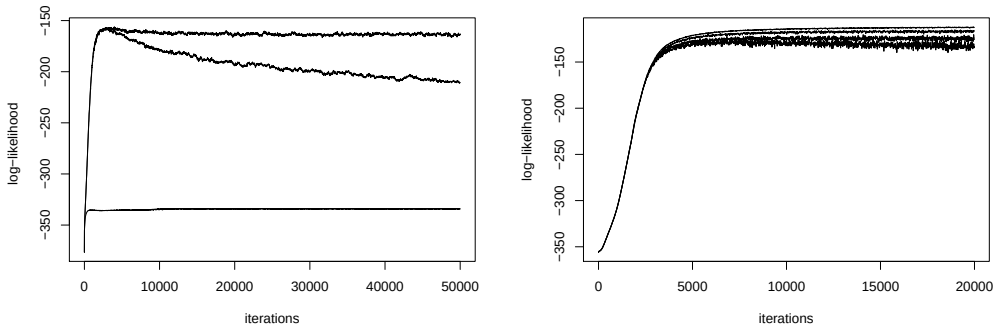


Figure 2.10: Evolution of the log-likelihood during training of an RBM on BAS. Left plot: with CD-1 and different values of the weight decay parameter (from bottom to top: $\lambda = 0.05, 0, 00005, 0, 0005$). Right Plot: with PT with different numbers M of temperatures (from bottom to top $M = 4, 5, 10, 50$). The values correspond to the medians over 25 runs.

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.

2.7.5 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 2.11 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.

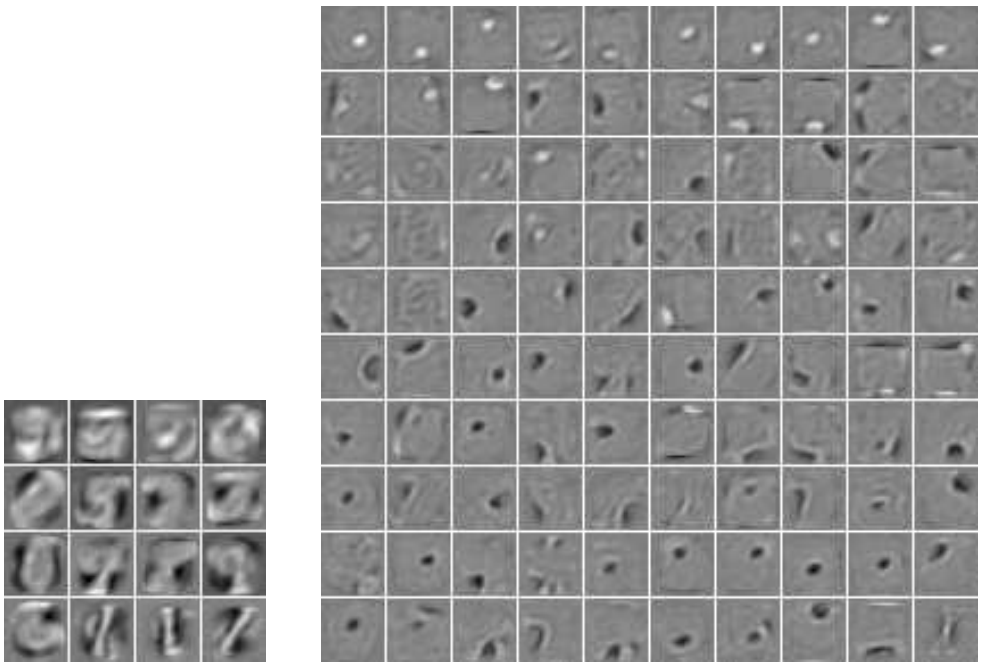


Figure 2.11: Visualization of the weights of RBMs with 16 and 100 hidden units (left and right plot, respectively) trained on MNIST. In the two plots, each square image shows the weights of one hidden unit. These images have the size of the input images, and each weight is visualized at the position of the corresponding visible unit. The gray values represent the size of the weights.

2.8 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., Taylor et al., 2007; Mnih et al., 2011). In these models, some of the parameters in the RBM energy are replaced by parametrized functions of some conditioning random variables, see the article by Bengio (2009) for an introduction. An obvious generalization is to remove the "R" from the RBM, which brings us back to the original Boltzmann machine (Ackley et al., 1985). 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 (Salakhutdinov and Hinton, 2009b).

The goal of this article was to introduce RBMs from the probabilistic graphical model perspective. It is meant to supplement existing tutorials (Bengio, 2009; Swersky

et al., 2010; Hinton, 2012), 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 the review by Bengio (2009) may serve as an excellent starting point.

Chapter 3

Empirical analysis of the divergence of Gibbs sampling based learning algorithms for RBMs

---

This chapter is based on the manuscript "Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines" by A. Fischer and C. Igel published in K. Diamantaras, W. Duch, and L. S. Iliadis, eds.: *International Conference on Artificial Neural Networks (ICANN)*, 6354 of LNCS, pp. 208-217. Springer, 2010.

Abstract

Learning algorithms relying on Gibbs sampling based stochastic approximations of the log-likelihood gradient have become a common way to train Restricted Boltzmann Machines (RBMs). We study three of these methods, Contrastive Divergence (CD) and its refined variants Persistent CD (PCD) and Fast PCD (FPCD). As the approximations are biased, the maximum of the log-likelihood is not necessarily obtained. Recently, it has been shown that CD, PCD, and FPCD can even lead to a steady decrease of the log-likelihood during learning. Taking artificial data sets from the literature we study these divergence effects in more detail. Our results indicate that the log-likelihood seems to diverge especially if the target distribution is difficult to learn for the RBM. The decrease of the likelihood can not be detected by an increase of the reconstruction error, which has been proposed as a stopping criterion for CD learning. Weight-decay with a carefully chosen weight-decay-parameter can prevent divergence.

3.1 Introduction

Training large undirected graphical models by vanilla likelihood maximization is in general computationally intractable because it involves averages over an exponential number of terms. Obtaining unbiased estimates of these averages by Markov chain Monte Carlo methods typically requires many sampling steps. However, biased estimates obtained after running a Gibbs chain for just a few steps can be sufficient for model training (Hinton, 2002). This is exploited by *Contrastive Divergence* (CD, Hinton, 2002) learning and its variants *Persistent CD* (PCD, Tieleman, 2008) and *Fast PCD* (FPCD, Tieleman and Hinton, 2009), which have been, for example, successfully applied to training of *Restricted Boltzmann Machines* (RBMs), the building blocks of *Deep Belief Networks* (DBNs, Hinton et al., 2006; Hinton and Salakhutdinov, 2006).

Contrastive Divergence learning is a biased approximation of gradient-ascent on the log-likelihood of the model parameters and thus does not necessarily reach the maximum likelihood estimate of the parameters. The bias depends on the mixing rate of the Markov chain, and mixing slows down with increasing model parameters (Hinton, 2002; Carreira-Perpiñán and Hinton, 2005; Bengio and Delalleau, 2009).[1] Recently it has been shown that the bias can lead to a divergence of the log-likelihood when training RBMs (Fischer and Igel, 2009; Desjardins et al., 2010b). In this study, we further investigate this divergence behavior and its dependence on the number of sampling steps used for the approximation, the number of hidden neurons of the RBM, and the choice of the weight decay parameter. After a brief description of CD, PCD, and FPCD, we describe our experiments, discuss the results and finally draw our conclusions.

3.2 Training RBMs

An RBM is an undirected graphical model (Hinton, 2002; Smolensky, 1986). Its structure is a bipartite graph consisting of one layer of visible units $\boldsymbol{V} = (V_1, \ldots, V_m)$ to represent observable data and one layer of hidden units $\boldsymbol{H} = (H_1, \ldots, H_n)$ to capture dependencies between observed variables. It is parametrized by the connection weights w_{ij} as well as the biases b_j and c_i of visible and hidden units, respectively ($i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$). Given these parameters, jointly denoted as $\boldsymbol{\theta}$, the modeled joint distribution of $\boldsymbol{V}$ and $\boldsymbol{H}$ is $p(\boldsymbol{v}, \boldsymbol{h}) = e^{-E(\boldsymbol{v},\boldsymbol{h})}/Z$, where $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ and the energy E is given by

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

[1] When referring to sizes of model parameters, we refer to their absolute values.

Differentiating the log-likelihood $\ell(\boldsymbol{\theta}|\boldsymbol{v}_l)$ of the model parameters $\boldsymbol{\theta}$ given one training example $\boldsymbol{v}_l$ with respect to $\boldsymbol{\theta}$ yields

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\boldsymbol{v}_l) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}_l) \frac{\partial E(\boldsymbol{v}_l, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad . \tag{3.1}$$

Computing the first term on the right side of the equation is straightforward because it factorizes. The computation of the second term is intractable for regular sized RBMs because its complexity is exponential in the size of the smallest layer. However, the expectation over $p(\boldsymbol{v})$ can be approximated by alternating Gibbs sampling (Ackley et al., 1985; Hinton and Sejnowski, 1986). But since the sampling chain needs to be long to get almost unbiased samples of the distribution modeled by the RBM, the computational effort is still too large.

Contrastive divergence. Instead of running the Gibbs chain until a near-to-equilibrium distribution is reached, in the k-step Contrastive Divergence (CD_k) algorithm (Hinton, 2002) the chain is run for only k steps, starting from an example $\boldsymbol{v}^{(0)}$ of the training set and yielding the sample $\boldsymbol{v}^{(k)}$. Each step t consists of sampling $\boldsymbol{h}^{(t)}$ from $p(\boldsymbol{h}|\boldsymbol{v}^{(t)})$ and sampling $\boldsymbol{v}^{(t+1)}$ from $p(\boldsymbol{v}|\boldsymbol{h}^{(t)})$ subsequently. The gradient (5.1) with respect to $\boldsymbol{\theta}$ of the log-likelihood for one training pattern $\boldsymbol{v}^{(0)}$ is then approximated by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad . \tag{3.2}$$

In the following, we restrict our considerations to RBMs with binary units for which $E_{p(h_i|\boldsymbol{v})}[h_i] = \mathrm{sig}\left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)$ with $\mathrm{sig}(x) = (1 + \exp(-x))^{-1}$.

The expectation $E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})}\left[\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)})\right]$ is denoted by $\mathrm{CD}_k^*(\boldsymbol{\theta}, \boldsymbol{v}^{(0)})$. Further, we denote the average of $\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)})$ over a training set by $\overline{\mathrm{CD}}_k(\boldsymbol{\theta})$ and its expectation by $\overline{\mathrm{CD}}_k^*(\boldsymbol{\theta})$. The expectations are considered for theoretical reasons. They lead to deterministic updates, but are computable only for small models.

Training RBMs using CD need not lead to a maximum likelihood estimate of the model parameters. Examples of energy functions and Markov chains for which CD_1 learning does not converge are given by MacKay (2001). Yuille (2005) specifies conditions under which CD learning is guaranteed to converge to the maximum likelihood solution, which need not hold for RBM training in general. Experiments comparing the quality of small RBMs trained based on CD_k^* and true likelihood maximization are presented in the analysis by Carreira-Perpiñán and Hinton (2005) and Bengio and Delalleau (2009).

Refined learning algorithms. More recently, refined algorithms also based on approximating the log-likelihood via Gibbs sampling have been proposed (Tieleman, 2008;

Tieleman and Hinton, 2009). In *Persistent Contrastive Divergence* (PCD, Tieleman, 2008) the sample $\boldsymbol{v}^{(k)}$ in the CD approximation (5.2) is sampled from a Markov chain defined by the RBM parameters that is independent of $\boldsymbol{v}^{(0)}$. This corresponds to standard CD learning without reinitializing the visible units of the Markov chain with the current training sample. It is assumed that the chain stays close to the stationary distribution if the learning rate is sufficiently small and thus the model changes only slightly between parameter updates (Younes, 1991; Tieleman, 2008). The PCD algorithm was further refined leading to a variant called *Fast Persistent Contrastive Divergence* (FPCD, Tieleman and Hinton, 2009). A set of additional parameters is introduced, which are only used for Gibbs sampling. The new parameters are referred to as *fast* parameters and should lead to higher mixing rates. When calculating the conditional distributions for Gibbs sampling, the regular parameters are replaced by the sum of the regular and the fast parameters. The update rule for the fast parameters is equal to that of the regular parameters, but with an independent, large learning rate and a large weight-decay parameter. Weight decay can also be used for the regular parameters, but it was suggested that regularizing just the fast weights is sufficient (Tieleman and Hinton, 2009). For details about (F)PCD we refer to the original publications (Tieleman, 2008; Tieleman and Hinton, 2009).

Limitations of the proposed learning algorithms. Bengio and Delalleau (2009) show that CD_k is an approximation of the true log-likelihood gradient by finding an expansion of the gradient that considers the k-th sample in the Gibbs chain and showing that CD_k is equal to a truncation of this expansion. Furthermore, they prove that the residual term (i.e., the bias of CD) converges to zero as k goes to infinity, and show empirically (by comparing the log-likelihood gradient and the expectation CD_k^* in small RBMs) that the quality of CD_k as an approximation of the log-likelihood gradient decreases as the norm of the parameters increases. Anyhow, the RBMs are still able to model the considered simple target distributions. Additionally, they find that the bias of CD_k also increases with increasing number of visible units.

Fischer and Igel (2009) show that CD can even lead to a steady decrease of the log-likelihood during learning. This is confirmed by Desjardins et al. (2010b) also for PCD and FPCD. Desjardins et al. (2010b) as well as Salakhutdinov (2009) further show that using algorithms based on tempered Markov chain Monte Carlo techniques yields better training results than Gibbs sampling.

3.3 Experiments

In our experiments, we study the evolution of the log-likelihood during gradient-based training of RBMs using CD_k, PCD, or FPCD. We first briefly describe our benchmark problems and then give details of the experimental setup.

Benchmark problems. We consider two artificial benchmark problems taken from the literature (Hinton and Sejnowski, 1986; MacKay, 2002). The *Labeled Shifter Ensemble* is a 19 dimensional data set containing 768 samples. The samples are generated in the following way: The states of the first 8 visible units are set uniformly at random. The states of the following 8 units are cyclically shifted copies of the first 8. The shift can be zero, one unit to the left, or one to the right and is indicated by the last three units. The log-likelihood is $768 \log \frac{1}{768} \approx -5102.43$ if the distribution of the data set is modeled perfectly.

Further, we consider a smaller variant of the *Bars-and-Stripes* problem described by MacKay (2002) with 16 instead of 25 visible units. Each pattern corresponds to a square of 4×4 units (if not stated otherwise) and is generated by first randomly choosing an orientation, vertical or horizontal with equal probability, and then picking the state for all units of every row or column uniformly at random. Since each of the two completely uniform patterns can be generated in two ways, the lower bound of the log-likelihood is -108.13.

Experimental setup. The RBMs were initialized with weights drawn uniformly from $[-0.5, 0.5]$ and zero biases. The number of hidden units was chosen to be equal to, twice, or half the number of the visible units.

The models were trained on both benchmark problems using gradient ascent on CD_k with $k \in \{1, 2, 4, 10, 100\}$, PCD, or FPCD. In the experiments presented here, we only discuss batch learning, but it was verified for CD that online learning leads to similar results. The batch update rule was augmented with optional weight-decay, that is, for CD_k we have

$$\boldsymbol{\theta}^{(g+1)} = \boldsymbol{\theta}^{(g)} + \eta \overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)}) - \lambda \boldsymbol{\theta}^{(g)} \quad . \tag{3.3}$$

We tested different learning rates η and values of the weight-decay parameter λ (which is set to zero if not stated otherwise). Using a momentum term did not improve the results in our experiments (not shown). For the fast parameters in FPCD the learning rate was set to 0.1 and the weight-decay-parameter was set to $\lambda_{\text{fast}} = \frac{19}{20}$ as suggested by Tieleman and Hinton (2009).

In order to analyze stochastic effects of the Gibbs sampling, we also did experiments using the computationally expensive expectation $\overline{\mathrm{CD}}_1^*(\boldsymbol{\theta}^{(g)})$ of the CD update on a

further reduced Bars-and-Stripes problem with 9 visible units.

To save computation time, the exact likelihood was calculated only every 10 iterations of the learning algorithm. We additionally computed the mean reconstruction error, which has been proposed as an early stopping criterion for CD training (Bengio et al., 2007; Taylor et al., 2007) and is typically used to train autoassociators (Bengio et al., 2007; Rumelhart et al., 1986a). The reconstruction error for one training example $\boldsymbol{v}$ is given by $-\log P\left(\boldsymbol{v}\middle|E\left[\boldsymbol{H}|\boldsymbol{v}\right]\right)$. All experiments were repeated 25 times and the medians of the results are presented if not stated otherwise.

Results. The evolution of the median log-likelihood for CD_1 with different learning rates is shown in Figure 3.1. After an initial increase, the log-likelihood steadily decreases and the model gets worse. This happens systematically in every trial as indicated by the quartiles. The higher the learning rate (i.e., the faster the learning) the more pronounced the divergence.

The decrease of the log-likelihood was also observed if the expectation $\overline{\mathrm{CD}}_1^*$ of the 1-step sample was used instead of a real sample. This shows that the observed effects are not caused by sampling noise. Because of computational complexity, we performed only three single trials with different learning rates on a smaller variant of the Bars-and-Stripes problem (3×3 pixel) in the $\overline{\mathrm{CD}}_1^*$ experiments, see right plot in Figure 3.2.

Without weight decay, the norm (we looked at both ∞- and 2-norm) of the RBM parameters steadily increased (this is no surprise and therefore the results are not shown).

Comparing the course of the log-likelihood with the corresponding evolution of the reconstruction error, also shown in Figure 3.1, reveals that the reconstruction error constantly decreased in our experiments. Thus, an *increase* of the reconstruction error could not be used as a criterion for early-stopping.

As shown in the left plot of Figure 3.2, using a decaying learning rate $\eta^{(g)}$ (with decay schedule $\eta^{(g)} = \frac{c_1}{c_2+g}$, where $c_1, c_2 \in \mathbb{R}^+$) can prevent divergence. However, if the learning rate decays too fast (c_2 is small), learning will become too slow and if $\eta^{(g)}$ decays too slowly, the divergence is still observed.

The plots in Figure 3.3 show the dependence on the number of sampling steps k. As expected, the larger k the less severe the divergence. However, in our experiments we observed a clear decrease of the likelihood even for $k = 10$. For $k = 100$, there was only a slight final decrease of the likelihood in the Shifter task and a negligible decrease in the Bars-and-Stripes benchmark.

The effect of L_2-norm weight-decay with different weight-decay parameters λ is shown in Figure 3.4. The results indicate that the choice of the hyperparameter λ is

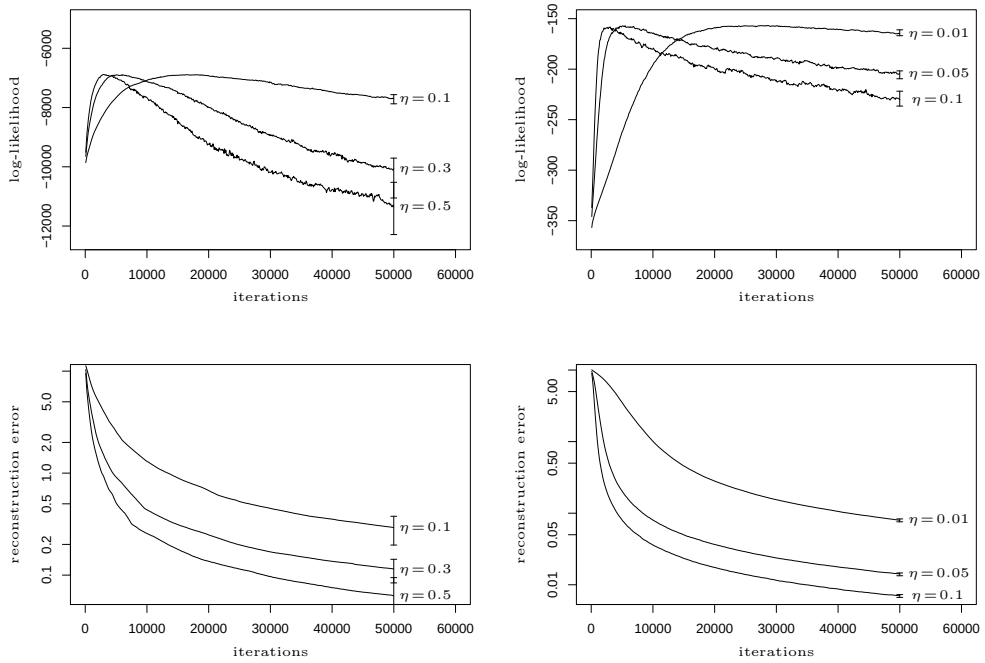


Figure 3.1: Top: Evolution of log-likelihood for CD_1 using steepest-descent with different learning rates. Shown are the medians over 25 trials for the Shifter (left) and Bars-and-Stripes (right) problem, error bars indicate quartiles. Bottom: Corresponding reconstruction error.

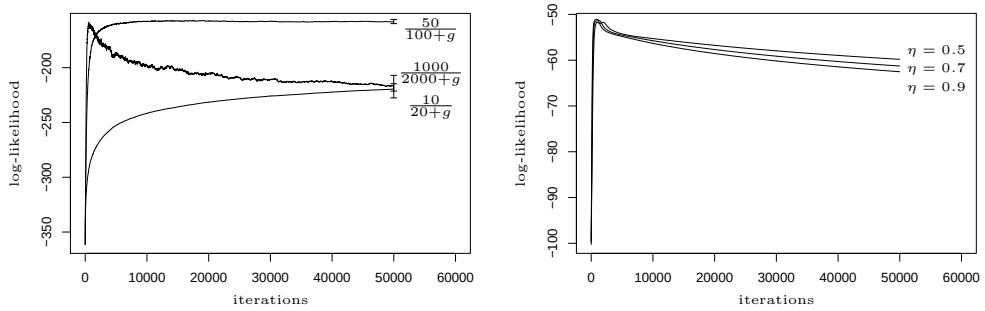


Figure 3.2: Left: Log-likelihood for CD_1 with different adaptive learning rates for Bars-and-Stripes. Right: Log-likelihood for CD_1^* for the smaller Bars-and-Stripes problem with 3×3 units. In contrast to all other plots, here only single trials and not medians are shown.

crucial. If it was chosen correctly, the divergence problem was solved. If λ was too large, the RBMs did not model the target distribution accurately. If the weight-decay parameter was too small, it could not prevent divergence.

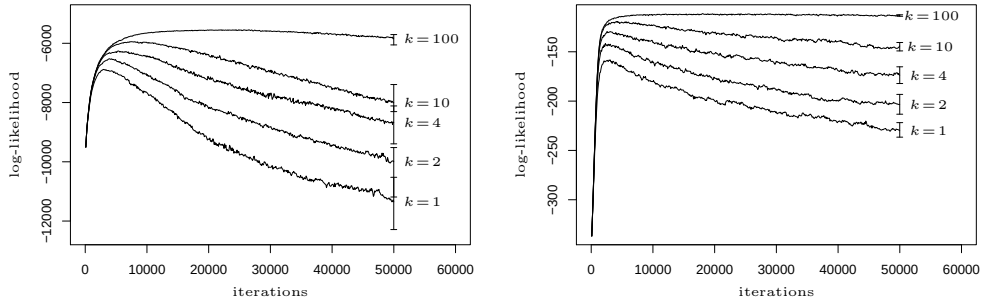


Figure 3.3: Log-likelihood for CD_k with different choices of k. The learning rates were $\eta = 0.5$ and $\eta = 0.1$ for the Shifter (left) and Bars-and-Stripes (right) problem.

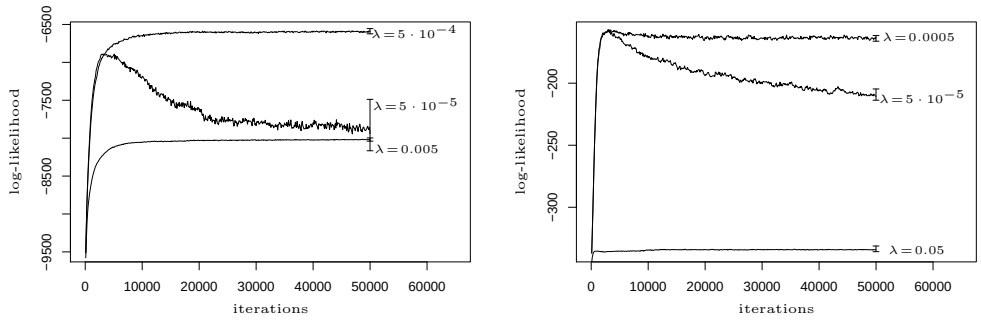


Figure 3.4: Evolution of the log-likelihood for CD_1 and weight-decay with different weight-decay parameters λ and learning rates as in Figure 3.3 for the Shifter (left) and Bars-and-Stripes (right) problem.

The influence of the number of hidden units is demonstrated in Figure 3.5. The more hidden units the more expressive power the RBM has (Le Roux and Bengio, 2008). Thus, the more hidden units the "easier" the problem for the RBM. Therefore, the results in Figure 3.5 suggest that the easier the problem the lower the risk of divergence.

The learning curves for PCD and FPCD were very similar to each other. We observed the same divergence effects (e.g., see Figure 3.6) that could be tamed by weight-decay (weight-decay results are not shown).

3.4 Discussion

We have shown on benchmark problems from the literature that vanilla gradient-based optimization of RBMs via k-step CD, PCD, and FPCD can lead to a systematic decrease of the likelihood after an initial increase. The reason for this is that an increase

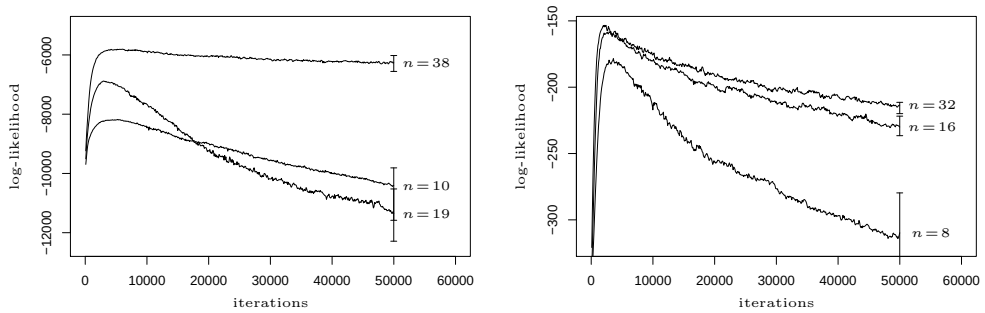


Figure 3.5: Log-likelihood for CD_1 applied to RBMs with different numbers of hidden variables n for the Shifter (left) and Bars-and-Stripes (right) problem.

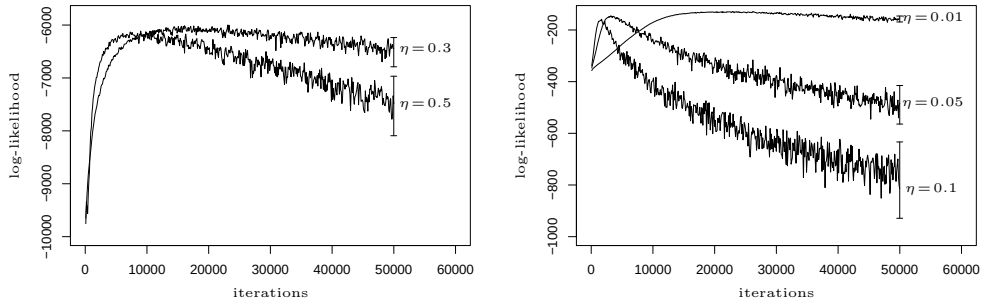


Figure 3.6: Evolution of log-likelihood for PCD (Shifter, left plot) and FPCD (Bars-and-Stripes, right plot) depending on learning rate η.

of the model parameters increases the difference between the CD update and a gradient step on the log-likelihood. The weight increase steadily slows down the mixing rate of the Gibbs chain associated with the CD approximation (the fact that Gibbs chains in general converge faster if the start distribution is close to the target distribution does not compensate for this). With increasing weights the mixing rate goes down and makes sampling in the Gibbs chain more and more deterministic, thus inducing a strong bias. If (for some components) this bias further increases the respective weights and decreases the likelihood, CD learning diverges. However, is this really a problem in practice? One may argue that there are several well-known and simple ways to address this problem, namely (i) early stopping of the learning process (Bengio et al., 2007; Taylor et al., 2007), (ii) regularization using weight-decay (Hinton, 2002; Hinton and Salakhutdinov, 2006; Hinton et al., 2006), and (iii) increasing k dynamically when the weights increase (Bengio and Delalleau, 2009).

Early stopping requires some reliable indicator that tells us when to stop and that can be computed efficiently. However, monitoring the true likelihood periodically is only possible for small problems, and the reconstruction error as discussed by Bengio

et al. (2007) and Taylor et al. (2007) may gradually decrease in spite of decreasing likelihood as, for example, shown in our experiments. An adaptive learning rate can have a similar effect as early-stopping. A decaying learning rate $\eta^{(g)}$ with an appropriate schedule can prevent divergence. However, choosing the right schedule is crucial and difficult. If the learning rate decays too fast, learning will become too slow and we will not reach sufficiently good solutions in time. If $\eta^{(g)}$ decays too slowly, the learning rate adaptation has little effect and divergence is still observed.

Weight decay offers a solution – if the regularization parameter is chosen correctly. If chosen too large, the model may not represent the target density accurately enough. If chosen too small, the decay term does not prevent divergence.

As suggested by Bengio and Delalleau (2009), increasing k can be a good strategy to find models with higher likelihood and it can also prevent divergence. However, divergence occurs even for values of k too large to be computationally tractable for large models. Thus, a dynamic schedule that enlarges k as the weights increase is needed (Bengio and Delalleau, 2009). Finding such a schedule is an open problem.

It seems that the more difficult the problem (i.e., the more difficult it is for the RBM to represent the target density) the more pronounced the divergence effect. The low-dimensional problems investigated by Bengio and Delalleau (2009) are all rather easy to learn for the considered RBMs and therefore the divergence is not apparent in that study. The dependence on difficulty makes the observed phenomenon relevant for DBNs. In these multi-layer architectures, simple models such as RBMs are used in each layer and the complexity of the target distribution is reached by stacking these simple models. The lower layer(s) cannot (or should not) represent the target density alone – and thus RBMs in DBNs face distributions that are difficult to learn.

3.5 Conclusion

Optimization based on k-step Contrastive Divergence (CD) or (Fast) Persistent CD is a promising approach to train undirected graphical models. It has proven to be successful in challenging applications and has contributed significantly to the current revival of Restricted Boltzmann Machines (RBMs) and deep architectures. The CD is a biased estimate of the desired update direction. This bias is reduced with increasing k and gets worse with increasing norm of the model parameters (i.e., slower mixing rates of the involved Markov chain). While it is common knowledge that CD learning may only approximate the maximum likelihood solution, we showed that the bias can lead to divergence of the learning process in the sense that the model systematically and drastically gets worse if k is not large. Thus, for training algorithms relying on Gibbs sampling based stochastic approximations of the log-likelihood gradient, there is

a need for robust mechanisms that control the weight growth in CD and related learning algorithms, for example, reliable heuristics for choosing the weight decay parameters or suitable criteria for early-stopping. New learning methods for RBMs using Markov chain Monte Carlo algorithms based on tempered transitions are promising (Salakhutdinov, 2009; Desjardins et al., 2010b), but their learning and scaling behavior needs to be further explored.

Chapter 4

Training RBMs based on the signs of the CD approximation of the log-likelihood derivatives

---

This chapter is based on the manuscript "Training RBMs based on the signs of the CD approximation of the log-likelihood derivatives" by A. Fischer and C. Igel published in M. Verleysen, ed.: *19th European Symposium on Artificial Neural Networks (ESANN)*, pp. 495-500, Belgium: d-side publications, 2011.

Abstract

Contrastive Divergence (CD) learning is frequently applied to Restricted Boltzmann Machines (RBMs), the building blocks of deep believe networks. It relies on biased approximations of the log-likelihood gradient. This bias can deteriorate the learning process. It was claimed that the signs of most components of the CD update are equal to the corresponding signs of the log-likelihood gradient. This suggests using optimization techniques only depending on the signs. Resilient backpropagation is such a method and we combine it with CD learning. However, it does not prevent divergence caused by the approximation bias.

4.1 Introduction

The rising field of deep learning led to a revival of Restricted Boltzmann Machines (RBMs, Smolensky, 1986) as typical building blocks of Deep Belief Networks (DBNs, e.g., see Bengio, 2009). Standard training of DBNs requires sequential training of RBMs in a layer wise fashion. Thus, effective and robust methods for RBM learning are a prerequisite for DBN training. Preforming maximum likelihood learning by vanilla steepest ascent is not possible in RBMs because the log-likelihood gradient involves averages which are exponential in the size of the smaller RBM layer and is thus not tractable. Therefore, *Contrastive Divergence* (CD, Hinton, 2002) learning has become the standard learning algorithm.

The k-step CD update is based on steepest ascent on a biased estimate of the log-likelihood gradient gained by k steps of Gibbs sampling. The bias of the approximation depends on the mixing rate of the Markov chain, and mixing slows down with increasing absolute value of the model parameters (Hinton, 2002; Carreira-Perpiñán and Hinton, 2005; Bengio and Delalleau, 2009; Fischer and Igel, 2011a). Hence, the bias increases with increasing parameter magnitude during RBM training. Recently, it has been shown that this can lead to divergence of the log-likelihood (Fischer and Igel, 2009; Desjardins et al., 2010b; Fischer and Igel, 2010a). Nevertheless, by comparing the log-likelihood gradient and the expectation of the CD-k update in small RBMs (where both are tractable), Bengio and Delalleau (2009) found that the fraction of parameter updates for which the log-likelihood derivatives and the corresponding CD-k components have different signs is small. Thus, optimization algorithms which consider only the signs of the partial derivatives could lead to better learning results.

The speed of steepest ascent CD learning crucially depends on the learning rate (see for example the empirical results from Fischer and Igel (2010a)). For large-scale problems, optimization algorithms are needed that increase the likelihood in few iterations without extensive hyperparameter tuning. As the likelihood is in general intractable, the choice of the gradient-based optimization algorithm is limited to methods that do not need the absolute objective function value.

Resilient backpropagation (RProp, Riedmiller, 1994; Igel and Hüsken, 2003) is a powerful learning algorithm frequently used for neural network training. It autonomously adapts the step sizes for the parameter updates during learning. The algorithm depends only on the signs of the partial derivatives of the objective function and not on their absolute values. The standard variant does also not require the value of the objective function. Thus, RProp appears to be the ideal learning algorithm for training RBMs based on approximations of the log-likelihood gradient. It could suffer less from approximation errors, because it just requires the signs, and can adapt the learning rate automatically. Therefore, the combination of RProp and CD

is empirically investigated in this paper. After briefly describing RBMs, CD-leaning, and RProp, we describe our experiments, discuss the results, and finally draw our conclusions.

4.2 RBMs and CD learning

An RBM is a bipartite undirected graphical model. The joint distribution of the m visible and n hidden (latent) variables under the model is $p(\boldsymbol{v}, \boldsymbol{h}) = e^{-E(\boldsymbol{v},\boldsymbol{h})} / \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$. The energy E is given by $E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{v} - \boldsymbol{v}^{\mathrm{T}}\boldsymbol{b} - \boldsymbol{h}^{\mathrm{T}}\boldsymbol{c}$ with parameters $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c})$, $\boldsymbol{W} \in \mathbb{R}^{n\times m}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^n$.

The basic idea of the k-step Contrastive Divergence (CD-k) algorithm (Hinton, 2002) is to run a Gibbs chain for only k steps starting from a training example $\boldsymbol{v}^{(0)}$ producing the sample $\boldsymbol{v}^{(k)}$. Each step t consists of sampling $\boldsymbol{h}^{(t)}$ from $p(\boldsymbol{h}|\boldsymbol{v}^{(t)})$ and sampling $\boldsymbol{v}^{(t+1)}$ from $p(\boldsymbol{v}|\boldsymbol{h}^{(t)})$ subsequently. The gradient with respect to $\boldsymbol{\theta}$ of the log-likelihood for $\boldsymbol{v}^{(0)}$ is then approximated by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad . \tag{4.1}$$

4.3 Training RBMs with resilient backpropagation

Resilient backpropagation is an iterative algorithm with adaptive individual step sizes (Riedmiller, 1994). It is frequently used for unconstrained optimization in machine learning because it is fast and robust with respect to the choice of the internal (hyper-) parameters, has linear time and space complexity in the number of parameters to be optimized, and is not very sensitive to numerical problems. The basic version of the RProp algorithm[1] considers only the signs of the partial derivatives of the function to be optimized and not their values. Because experiments suggest that the CD estimator has the correct sign most of the time (Bengio and Delalleau, 2009), RProp seems to be promising for CD learning.

Let the ith component of the mean CD-approximation over the training set be denoted by $[\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i$. In each iteration g of RProp, every parameter $\theta_i^{(g)}$ is updated according to

$$\theta_i^{(g+1)} = \theta_i^{(g)} + \operatorname{sign}\left([\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i\right) \cdot \Delta_i^{(g)} \quad . \tag{4.2}$$

Prior to this update, the step size $\Delta_i^{(g)}$ is adapted based on changes of sign of the (approximated) partial derivative $[\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i$ in consecutive iterations. If the sign

[1] Usually we prefer the RProp variant called iRProp$^+$ (Igel and Hüsken, 2003). However, this method requires the value of the objective function and not just the partial derivatives. Therefore, it is not well suited for the problem at hand.

changes, which indicates that a local minimum has been overstepped, then the step size is multiplicatively decreased, otherwise, it is increased. The update rule for the step size is given by:

$$\Delta_i^{(g)} = \begin{cases} \min(\eta^+ \Delta_i^{(g-1)}, \Delta_{\max}) & \text{if } [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g-1)})]_i [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i > 0 \\ \max(\eta^- \Delta_i^{(g-1)}, \Delta_{\min}) & \text{if } [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g-1)})]_i [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i < 0 \\ \Delta_i^{(g-1)} & \text{otherwise} \ , \end{cases} \tag{4.3}$$

where $0 < \eta^- < 1 < \eta^+$ and the step size is bounded by $\Delta_{\min}$ and $\Delta_{\max}$.

4.4 Experiments

We considered four artificial benchmark problems taken from the literature (MacKay, 2002; Bengio and Delalleau, 2009; Fischer and Igel, 2010a). The *Labeled Shifter Ensemble* is a 19 dimensional data set containing 768 different samples and so the log-likelihood is $768 \log \frac{1}{768} \approx -5102.43$ if the distribution of the data set is modeled perfectly. We consider a variant of the *Bars and Stripes Ensemble* with 16 units and 32 input pattern yielding a bound of the log-likelihood of -108.13. Furthermore we considered the Diag_d-problem and the $\mathit{1DBall}_d$-problem with $d = 6$ dimensions. The first data set contains 7, the latter 24 unique binary vectors. The bounds for the log-likelihood are -13.62 and -76.27, respectively.

The RBMs were initialized with weights drawn uniformly from $[-0.5, 0.5]$ and zero biases. The numbers of hidden units were chosen to be equal to the number of visible units. The models were trained with RProp based on CD-1 or CD-100 on all four benchmark problems (batch learning). If not stated otherwise, the hyperparameters where set to the default values $\eta^- = 0.5$, $\eta^+ = 1.1$, $\Delta_{\min} = 0.0$ and $\Delta_{\max} = 10^{100}$. To save computation time, the exact likelihood was calculated only every 10 iterations of the learning algorithm. All experiments were repeated 25 times.

4.5 Results

The left plot of figure 4.1 shows the evolution of the log-likelihood during learning of the Shifter-problem with RProp based on CD-1. After an increase in the first iterations the log-likelihood starts to decrease. The development of the likelihood differs a lot depending on the parameter initialization as can be seen exemplarily in the inset plot depicting some single trials. When using the CD-100 instead of the CD-1 approximation of the gradient a stagnation of the log-likelihood on an unsatisfying level during RProp based learning is observed (see right plot of Figure 4.1). This happens systematically

in every trail independent of the initialization of the parameters as indicated by the quartiles.

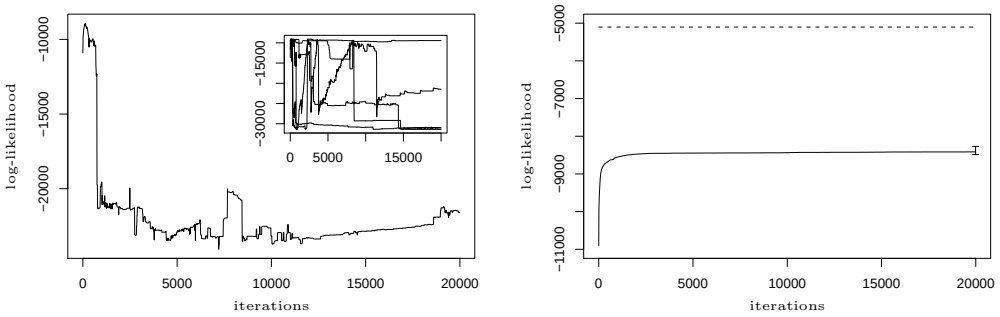


Figure 4.1: The development of the log-likelihood during training RBMs on the Shifter-problem with RProp. Shown are the medians over 25 trails. Left: RProp based on CD-1. Five single trails with different parameter initializations are exemplarily shown in the inset plot. Right: RProp based on CD-100. Error bars indicate quartiles, the dashed line indicates the upper bound of the log-likelihood.

During training an RBM on the Bars-and-Stripes-problem the log-likelihood stagnates when RProp is based on CD-1 as well as on CD-100. In both settings similar log-likelihood values are reached which are low compared to the upper bound and to the maximum values reached when an RBM is trained with steepest ascent (see empirical analysis by Fischer and Igel (2010a)).

The log-likelihood also stagnates when learning the Diag- and 1DBall-problem. Here we also observe similar learning curves if the RProp algorithm is based on CD-1 and CD-100. The results are not shown due to the great similarity to the right plot of Figure 4.1. For further empirical results we refer to the accompanying technical report (Fischer and Igel, 2010b).

The stagnation of the log-likelihood could indicate a frequently changing sign of the CD-approximation during the learning process. A frequently changing sign of the approximation causes the step size for the parameter updates (4.3) to get smaller and smaller and – if the step size is not bounded – to finally approach zero. Thus it could be possible to avoid the stagnation by enlarging the minimal possible step size $\Delta_{\min}$. This idea is verified by the following results.

If the minimal step size $\Delta_{\min}$ is set to a value larger than zero and the maximal step size $\Delta_{\max}$ is set to a value smaller than the default value, we can observe big differences in the evolution of the log-likelihood during RProp based training. As shown in Figure 4.2, high (relative to the upper bound) log-likelihood values are reached during learning the Diag- and the 1DBall-problem with RProp based on CD-1 if the

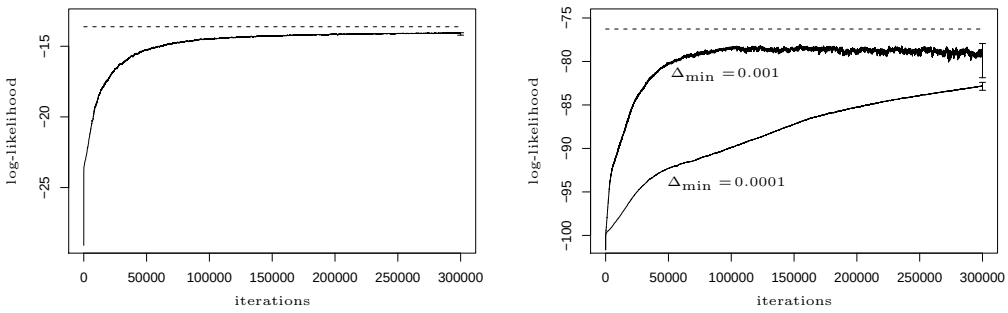


Figure 4.2: Log-likelihood during training with RProp with limited step size hyperparameters. On the left: Learning of the Diag-problem. The step size is limited by $\Delta_{\min} = 0.0001$ and $\Delta_{\max} = 1$. On the right: Learning of the 1DBall-problem with $\Delta_{\max} = 1$ and $\Delta_{\min} = 0.001$ and $\Delta_{\min} = 0.0001$ respectively.

hyperparameters are set to $\Delta_{\min} = 0.0001$ or $\Delta_{\min} = 0.001$, respectively, and $\Delta_{\max} = 1$. A nearly identical evolution of the log-likelihood can be observed (results not shown) if only the minimal step size value is enlarged and $\Delta_{\max}$ is set to its default value.

When learning the Bars-and-Stripes-problem with a restriction of the step size parameters ($\Delta_{\min} = 0.0001$ and $\Delta_{\max} = 1$) the log-likelihood starts to diverge. The experiments with the Shifter-problem with restricted step size parameters lead to similar results as the experiments with the parameters set to the default values.

4.6 Discussion and conclusion

The experiments show that the success of RProp based CD learning depends on the data distribution to be learned and on the values of the hyperparameters ($\Delta_{\min}$ and $\Delta_{\max}$). If the step size is allowed to get arbitrary close to zero ($\Delta_{\min} = 0.0$), the training progress stagnated on an unsatisfying level for some target distribution. With an appropriate $\Delta_{\min}$, RProp was able to learn good models depending on the problem.

The reason for the stagnation could be convergence to a suboptimal local maximum. However, experiments using the expectation of CD-1 (not shown) did not suffer from the stagnation problem. As we see no reason why learning based on the expectation of CD-1 should be less prone to getting stuck in undesired local optima than learning based on the CD approximation, local maxima are not likely to be the reason.

We believe that the reason is the fast reduction of the RProp step size parameters Δ_i due to changes in sign of the gradient components induced by stochastic effects and errors in the CD approximation. Frequent changes of the sign could also be caused by ill-shaped log-likelihood functions (Igel and Hüsken, 2003). However, if the RBMs were trained with RProp based on the exact likelihood gradient, good models for all

benchmark problems could be learned in few iterations (results not shown).

If steepest ascent can learn a distribution reliably (e.g., when applied to Diag_6 or 1DBall_6), this is also possible using RProp, but in our experiments this required $\Delta_{\min} > 0$. In these cases, RProp may be preferable because $\Delta_{\min}$ may be easier to tune than learning rates in steepest ascent.

When applying RProp to Shifter and Bars-and-Stripes, the log-likelihood diverged before a good model was learned even if we constrained $\Delta_{\min}$ and $\Delta_{\max}$. That is, if learning diverges using steepest ascent (as reported by Fischer and Igel (2010a)), it also diverged using RProp. Thus, albeit it has been reported that the sign of the components of the CD update direction vector is often right, learning based on these signs tends to diverge.

In future work, we will evaluate RProp in combination with other approximations of the log-likelihood gradient (e.g., based on tempered transitions (Desjardins et al., 2010b)).

Chapter 5

Bounding the bias of contrastive divergence learning

This chapter is based on the manuscript "Bounding the bias of contrastive divergence learning" by A. Fischer and C. Igel published in *Neural Computation* 23, pp. 664-673, 2011.

Abstract

Optimization based on k-step Contrastive Divergence (CD) has become a common way to train Restricted Boltzmann Machines (RBMs). The k-step CD is a biased estimator of the log-likelihood gradient relying on Gibbs sampling. We derive a new upper bound for this bias. Its magnitude depends on k, on the number of variables in the RBM, and on the maximum change in energy that can be produced by changing a single variable. The latter reflects the dependence on the absolute values of the RBM parameters. The magnitude of the bias is also affected by the distance in variation between the modeled distribution and the starting distribution of the Gibbs chain.

5.1 Training RBMs using contrastive divergence

Restricted Boltzmann Machines (RBMs) are undirected graphical models (Smolensky, 1986; Hinton, 2002). The RBM structure is a bipartite graph consisting of one layer of observable variables $\boldsymbol{V} = (V_1, \ldots, V_m)$ and one layer of hidden (latent) variables $\boldsymbol{H} = (H_1, \ldots, H_n)$. The modeled distribution is given by $p(\boldsymbol{v}, \boldsymbol{h}) = e^{-E(\boldsymbol{v},\boldsymbol{h})}/Z$ where $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ and the energy E is given by

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

with real-valued parameters w_{ij}, b_j, and c_i ($i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$, and $w_{ij} = w_{ji}$) jointly denoted as $\boldsymbol{\theta}$. In the following, we restrict our considerations to RBMs with binary units for which $E_{p(h_i|\boldsymbol{v})}[h_i] = \mathrm{sig}\left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)$ with $\mathrm{sig}(x) = (1 + \exp(-x))^{-1}$.

Differentiating the log-likelihood $\ell(\boldsymbol{\theta}|\boldsymbol{v}_l)$ of the model parameters $\boldsymbol{\theta}$ given one training example $\boldsymbol{v}_l$ with respect to $\boldsymbol{\theta}$ yields

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\boldsymbol{v}_l) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}_l) \nabla_{\boldsymbol{\theta}} E(\boldsymbol{v}_l, \boldsymbol{h}) + \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \nabla_{\boldsymbol{\theta}} E(\boldsymbol{v}, \boldsymbol{h}) \ . \tag{5.1}$$

Computing the first term on the right side of the equation is straightforward because it factorizes nicely. The computation of the second term is intractable for regular sized RBMs because its complexity is exponential in the size of the smallest layer.

Therefore, k-step Contrastive Divergence (CD-k) learning (Hinton, 2002) approximates the second term by a sample obtained by k-steps of Gibbs sampling. Starting from an example $\boldsymbol{v}^{(0)}$ of the training set, the Gibbs chain is run for only k steps yielding the sample $\boldsymbol{v}^{(k)}$. Each step t consists of sampling $\boldsymbol{h}^{(t)}$ from $p(\boldsymbol{h}|\boldsymbol{v}^{(t)})$ and sampling $\boldsymbol{v}^{(t+1)}$ from $p(\boldsymbol{v}|\boldsymbol{h}^{(t)})$ subsequently. The gradient (5.1) with respect to $\boldsymbol{\theta}$ of the log-likelihood for one training pattern $\boldsymbol{v}^{(0)}$ is then approximated by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \nabla_{\boldsymbol{\theta}} E(\boldsymbol{v}^{(0)}, \boldsymbol{h}) + \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \nabla_{\boldsymbol{\theta}} E(\boldsymbol{v}^{(k)}, \boldsymbol{h}) \ . \tag{5.2}$$

Bengio and Delalleau (2009) show that CD-k is an approximation of the true log-likelihood gradient by finding an expansion of the gradient that considers the k-th sample in the Gibbs chain and showing that CD-k is equal to a truncation of this expansion. Furthermore, they prove that the left out term converges to zero as k goes to infinity:

Theorem 5.1 (Bengio and Delalleau, 2009, p. 1608)**.** *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*

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \nabla_{\boldsymbol{\theta}} E(\boldsymbol{v}^{(0)}, \boldsymbol{h}) + E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \nabla_{\boldsymbol{\theta}} E(\boldsymbol{v}^{(k)}, \boldsymbol{h}) \right] + E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{v}^{(k)}) \right]$$

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

In addition, Bengio and Dellalleau deduce a bound for the final term, which we refer to as the *bias of CD-k*, relating it to the magnitude of the RBM parameters:

$$\left| E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\frac{\partial \log p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] \right| \leq 2^m (1 - 2^m 2^n \operatorname{sig}(-\alpha)^m \operatorname{sig}(-\beta)^n)^k \ . \tag{5.3}$$

Here θ_a denotes a single parameter of the RBM, $\alpha = \max_j \left(\sum_i |w_{ij}| + |b_j| \right)$, $\beta = \max_i \left(\sum_j |w_{ij}| + |c_i| \right)$. But the bound gets loose very fast if the norm of the parameters increases. Note that the absolute value of

$$E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\frac{\partial \log p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] = \sum_{\boldsymbol{v}} p(\boldsymbol{v}^{(k)} = \boldsymbol{v}|\boldsymbol{v}^{(0)}) \frac{\partial \log p(\boldsymbol{v})}{\partial \theta_a}$$

is never larger than one for binary RBMs (this follows from $|\partial \log p(\boldsymbol{v})/\partial \theta_a| \leq 1$, e.g., see Bengio and Delalleau, 2009, p. 1611 above equation 4.3), while the bound given above grows quickly with α and β and approaches 2^m, the number of configurations of the visible units.

5.2 Bounding the CD approximation error

In this section, we derive a tighter bound for the bias in CD-k based on general results for the convergence rate of the Gibbs sampler (see Brémaud, 1999). The convergence rate depends on the distance between the distribution of the initial states μ (the starting distribution of the chain) and the stationary distribution. A measure of distance between two distributions α and β on a countable set Ω is the total variation distance defined as

$$d_V(\alpha, \beta) = \frac{1}{2} \|\alpha - \beta\|_1 = \frac{1}{2} \sum_{i \in \Omega} |\alpha(i) - \beta(i)| \ .$$

The total variation distance between two distributions is bounded by one. We make use of the following theorem:

Theorem 5.2. *Given a Markov random field* $\boldsymbol{X} = (X_1, ..., X_n)$ *with random variables taking values in a finite set* Ω *and a Markov chain* $(\boldsymbol{X}^{(k)})_{k \geq 0}$ *produced by periodic*

Gibbs sampling. Let $\mathbf{T}$ *be the transition matrix,* μ *the starting distribution, and* p *the stationary distribution (i.e., the Gibbs distribution) of the Gibbs chain. It holds*

$$\|\mu\mathbf{T}^k - p\|_1 \leq \frac{1}{2}\|\mu - p\|_1(1 - e^{-N\triangle})^k \quad , \tag{5.4}$$

where

$$\triangle = \sup_{l\in\{1,\ldots,n\}} \{|\mathcal{E}(\boldsymbol{x}) - \mathcal{E}(\boldsymbol{y})| \,;\, \boldsymbol{x},\boldsymbol{y} \in \Omega^n \text{ and } \forall i \in \{1,\ldots,n\} \setminus \{l\} : x_i = y_i\}$$

and $\mathcal{E}$ *denotes the energy function of the Gibbs distribution.*

A proof is given by Brémaud (1999, section 6.2, p. 289).

In the case of an RBM with hidden variables $\boldsymbol{H}$ and the visible variables $\boldsymbol{V}$ fixed to a pattern $\boldsymbol{v}^{(0)}$, the joint starting distribution is given by

$$\mu(\boldsymbol{v},\boldsymbol{h}) = \begin{cases} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) & \text{if } \boldsymbol{v} = \boldsymbol{v}^{(0)} \\ 0 & \text{otherwise} \end{cases} . \tag{5.5}$$

Now we can state our main result.

Theorem 5.3. *Given an RBM* $(V_1,\ldots,V_m,H_1,\ldots,H_n)$ *and a Markov chain produced by periodic Gibbs sampling starting from* $\boldsymbol{v}^{(0)}$ *(*$\boldsymbol{v}^{(0)} \Rightarrow \boldsymbol{h}^{(0)} \Rightarrow \boldsymbol{v}^{(1)} \Rightarrow \boldsymbol{h}^{(1)} \ldots$*). Let the initial states* $(\boldsymbol{v}^{(0)},\boldsymbol{h}^{(0)})$ *be distributed according to* μ *as defined in Eq.* (5.5) *and let* p *be the joint probability distribution of* $\boldsymbol{V}$ *and* $\boldsymbol{H}$ *of the RBM (i.e., the stationary distribution of the Markov chain). Then we can bound the error of the CD-k approximation of the log-likelihood derivative w.r.t some RBM parameter* θ_a *(i.e.,* $\partial\ell(\boldsymbol{\theta}|\boldsymbol{v}^{(0)})/\partial\theta_a$*) by*

$$\left| E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})}\left[\frac{\partial \log p(\boldsymbol{v}^{(k)})}{\partial\theta_a}\right]\right| \leq \frac{1}{2}\|\mu - p\|_1\left(1 - e^{-(m+n)\Delta}\right)^k \leq \left(1 - e^{-(m+n)\Delta}\right)^k$$

with

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

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

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

Proof. Bengio and Delalleau (2009) show

$$E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})}\left[\frac{\partial \log p(\boldsymbol{v}^{(k)})}{\partial\theta_a}\right] = \sum_{\boldsymbol{v}} \left(p(\boldsymbol{v}^{(k)} = \boldsymbol{v}|\boldsymbol{v}^{(0)}) - p(\boldsymbol{v})\right) \frac{\partial \log p(\boldsymbol{v})}{\partial\theta_a}$$

and use the inequality

$$\left| E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\frac{\partial \log p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] \right| \leq \sum_{\boldsymbol{v}} \left| \left(p(\boldsymbol{v}^{(k)} = \boldsymbol{v}|\boldsymbol{v}^{(0)}) - p(\boldsymbol{v}) \right) \right| \left| \frac{\partial \log p(\boldsymbol{v})}{\partial \theta_a} \right| \quad .$$

Instead of upper bounding the right hand side of this equation by

$$\max_{\boldsymbol{v}} \left| \left(p(\boldsymbol{v}^{(k)} = \boldsymbol{v}|\boldsymbol{v}^{(0)}) - p(\boldsymbol{v}) \right) \right| \left[2^m \max_{\boldsymbol{v}} \frac{\partial \log p(\boldsymbol{v})}{\partial \theta_a} \right]$$

as in the proof by Bengio and Delalleau (2009, equation 3.5), we bound it by

$$\begin{aligned}
&\sum_{\boldsymbol{v}} \left| \left(p(\boldsymbol{v}^{(k)} = \boldsymbol{v}|\boldsymbol{v}^{(0)}) - p(\boldsymbol{v}) \right) \right| \left| \frac{\partial \log p(\boldsymbol{v})}{\partial \theta_a} \right| \\
&\qquad = \sum_{\boldsymbol{v}} \left| \sum_{\boldsymbol{h}} \left(p^{(k)}(\boldsymbol{v}, \boldsymbol{h}) - p(\boldsymbol{v}, \boldsymbol{h}) \right) \right| \left| \frac{\partial \log p(\boldsymbol{v})}{\partial \theta_a} \right| \\
&\leq \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \left| \left(p^{(k)}(\boldsymbol{v}, \boldsymbol{h}) - p(\boldsymbol{v}, \boldsymbol{h}) \right) \right| \left| \frac{\partial \log p(\boldsymbol{v})}{\partial \theta_a} \right| \leq \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \left| \left(p^{(k)}(\boldsymbol{v}, \boldsymbol{h}) - p(\boldsymbol{v}, \boldsymbol{h}) \right) \right| \quad .
\end{aligned}$$

Here we use the notation $p^{(k)}(\boldsymbol{v}, \boldsymbol{h}) = p(\boldsymbol{v}^{(k)} = \boldsymbol{v}, \boldsymbol{h}^{(k)} = \boldsymbol{h}|\boldsymbol{v}^{(0)})$ and the fact that in binary RBMs we have $\left| \frac{\partial \log P(x)}{\partial \theta_a} \right| \leq 1$. The RHS is twice the total variation distance between the distribution of the variables of the RBM after k steps of Gibbs sampling and the stationary distribution of the chain.

Now we can apply Theorem 5.2 and get

$$\begin{aligned}
&\left| E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})} \left[\frac{\partial \log p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] \right| \leq \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \left| p^{(k)}(\boldsymbol{v}, \boldsymbol{h}) - p(\boldsymbol{v}, \boldsymbol{h}) \right| \\
&\leq \frac{1}{2} \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} |(\mu(\boldsymbol{v}, \boldsymbol{h}) - p(\boldsymbol{v}, \boldsymbol{h}))| \left(1 - e^{-(m+n)\Delta} \right)^k = \frac{1}{2} \|\mu - p\|_1 \left(1 - e^{-(m+n)\Delta} \right)^k \quad .
\end{aligned}$$

Here Δ denotes the maximum change in energy that can be produced by changing a single variable. We distinguish the two cases whether the maximum change is produced by a hidden or visible variable and define $\Delta = \max\{\Delta_v, \Delta_h\}$ using $\Delta_v = \max_{l \in \{1,\ldots,m\}} \vartheta_l$ and $\Delta_h = \max_{l \in \{1,\ldots,n\}} \xi_l$. For the visible units, we have

$$\vartheta_l = \max \{|E(\boldsymbol{v}, \boldsymbol{h}) - E(\boldsymbol{v}', \boldsymbol{h})|\} \quad ,$$

where we maximize over $\boldsymbol{v}'$, $\boldsymbol{v}$, and $\boldsymbol{h}$ under the constraint that $\forall j \in \{1, \ldots, m\}, j \neq$

$l : v_j = v'_j$ (i.e., that only one unit changes its state). Thus

$$\begin{aligned}
\vartheta_l &= \max\left\{\left| -\sum_{i=1}^{n}\sum_{j=1}^{m} h_i w_{ij} v_j - \sum_{j=1}^{m} v_j b_j - \sum_{i=1}^{n} h_i c_i \right.\right. \\
&\quad \left.\left. - \left(-\sum_{i=1}^{n}\sum_{j=1}^{m} h_i w_{ij} v'_j - \sum_{j=1}^{m} v'_j b_j - \sum_{i=1}^{n} h_i c_i \right) \right|\right\} \\
&= \max\left\{\left| \sum_{i=1}^{n} h_i w_{il}(v'_l - v_l) + b_l(v'_l - v_l) \right|\right\} \\
&= \max\left\{\left| \sum_{i=1}^{n} h_i w_{il} + b_l \right|\right\} \\
&= \max\left\{\left| \sum_{i=1}^{n} I_{\{w_{il}>0\}} w_{il} + b_l \right|, \left| \sum_{i=1}^{n} I_{\{w_{il}<0\}} w_{il} + b_l \right|\right\} \quad ,
\end{aligned}$$

where the indicator function I is one if its argument is true and zero otherwise. The third equality holds because $(v'_l - v_l)$ is either -1 or 1 and can be pulled out as a common factor. The absolute value of the resulting sum is maximized if the h_i exclusively "select" either all positive or all negative w_{il}, which leads to the final expression. Analogously, we compute ξ_s. □

The result for a single initial observed pattern $\boldsymbol{v}^{(0)}$ is appropriate for online learning. It is straight-forward to extend the theorem to batch learning, in which the gradient and the CD-k approximation are averages over a set of observed patterns defining an empirical distribution.

Corollary 5.1. *Let p denote the marginal distribution of the visible units of an RBM and let p_e 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*

$$\left| E_{p_e(\boldsymbol{v}^{(0)})}\left[E_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})}\left[\frac{\partial \log p(\boldsymbol{v}^{(k)})}{\partial \theta_a} \right] \right] \right| \leq \frac{1}{2}\|p_e - p\|_1 \left(1 - e^{-(m+n)\Delta}\right)^k$$

with Δ as defined in Theorem 5.3.

We can use p_e instead of the joint starting distribution $\mu(\boldsymbol{v}, \boldsymbol{h}) = p(\boldsymbol{h}|\boldsymbol{v})p_e(\boldsymbol{v})$ on the RHS of the equation because

$$\sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} |(\mu(\boldsymbol{v}, \boldsymbol{h}) - p(\boldsymbol{v}, \boldsymbol{h}))| = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \, |(p_e(\boldsymbol{v}) - p(\boldsymbol{v}))| = \sum_{\boldsymbol{v}} |(p_e(\boldsymbol{v}) - p(\boldsymbol{v}))| \quad .$$

Our bounds shows that the bias is determined by two antagonistic terms. The dependence on $\|p_e - p\|_1$ is an important difference between our results and those derived by Bengio and Delalleau (2009). Since p_e is the target distribution for the

RBM learning process, the variation distance between p_e and p should decrease during successful RBM learning. At the same time, the magnitudes of the parameters – if not controlled by weight decay – tend to increase in practice (see, e.g., Bengio and Delalleau, 2009; Fischer and Igel, 2009; Desjardins et al., 2010b). Thus Δ increases and $(1 - e^{-(m+n)\Delta})^k$ approaches one.

5.3 Experimental results

We empirically studied the development of bias and bound during learning of the Diag- and the 1DBall-dataset described by Bengio and Delalleau (2009, pp. 1613–1614). Small RBMs with 6 visible and 6 hidden neurons were trained via batch learning based on the expected value of the CD-1-update. Their parameters were initialized with weights drawn uniformly from $[-0.5, 0.5]$ and bias parameters set to $c_i = b_j = 0$ for all i and j. Each experiment was repeated 25 times with different initializations. The learning rates were set to 0.1 and no weight decay was used. The results are shown in Figure 5.3. The bias value plotted is the maximum $\max_{\theta_a} \left| E_{p(\boldsymbol{v}^{(1)}|\boldsymbol{v}^{(0)})} \left[\partial \log p(\boldsymbol{v}^{(1)}) / \partial \theta_a \right] \right|$ over all parameters.

The results show the tightness of the new bound. Only in the initial phase of learning, when $\|p_e - p\|_1$ is large, the bound was rather loose (but always non-trivial, i.e., below one, this is not shown in the left plot). After 50000 iterations the differences between bound and bias value as defined above are ≈ 0.00138 and ≈ 0.02628 for Diag and 1DBall, respectively. In the beginning, the bias is small because the models with weights close to zero mix fast (if the weights were all zero, the RBM would model a uniform distribution, which is sampled after a single Gibbs sampling step). We refer to the article by Bengio and Delalleau (2009) for a detailed empirical analysis of CD-k learning of RBMs applied to the Diag and 1DBall benchmark (e.g., showing the dependence on k and the dimensionality of the problem).

In comparison the previously described bound given in (5.3) exceeds the value of 1 already for $\alpha \geq 0.003$ and $\beta \geq 0.003$ for an RBM with 6 visible and 6 hidden variables and $k = 1$ as used in the experiments. Values of this size are reached for α and β either immediately after initialization of the parameters or after a small number of training iterations, and the bound grows fast with increasing magnitude of the parameters (e.g. for values $\alpha, \beta \geq 1$ it is already bigger than 63).

5.4 Discussion and conclusion

We derived a new upper bound for the bias when estimating the log-likelihood gradient by k-step Contrastive Divergence (CD-k) for training RBMs. It is considerably tighter

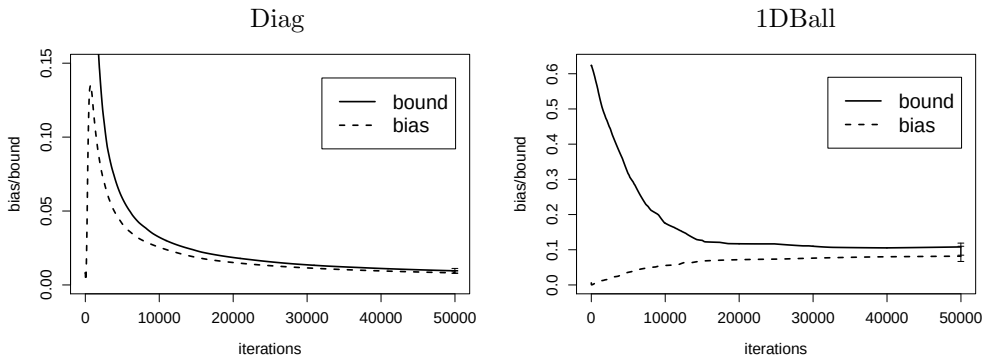


Figure 5.1: The development of bias and bound during learning of the Diag- (left) and the 1DBall-dataset (right) described by Bengio and Delalleau (2009). The bias value plotted is the maximum $\max_{\theta_a} \left| E_{p(\boldsymbol{v}^{(1)}|\boldsymbol{v}^{(0)})} \left[\partial \log p(\boldsymbol{v}^{(1)}) / \partial \theta_a \right] \right|$ over all parameters. Shown are the medians over 25 trials, error bars indicate quartiles.

than a recently published result. The main reason for that is that it incorporates the decrease of the bias for decreasing distance between the modeled distribution and the starting distribution of the Gibbs chain.

Learning based on CD-k has been successfully applied to RBMs (e.g., Hinton, 2002; Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Bengio et al., 2007; Hinton, 2007a; Bengio and Delalleau, 2009). However, it need not converge to a maximum likelihood (ML) estimate of the model parameters (conditions for convergence with probability one are given by Yuille (2005)). Analytical counterexamples are presented by MacKay (2001). Carreira-Perpiñán and Hinton (2005) show that in general CD learning does not lead to the ML solution. In their experiments it reaches solutions close by. However, empirical evidences for misled RBM learning using approximations of the true log-likelihood gradient are, for example, given by Fischer and Igel (2009; 2010a) as well as Desjardins et al. (2010b). Intuitively, the smaller the bias of the log-likelihood gradient estimation, the higher the chances to converge to an ML solution quickly. Still, even small deviations of a few gradient components can deteriorate the learning process.

Our bound for the bias increases with the maximum possible change in energy that can be produced by changing a single variable. This indicates the relevance of controlling the absolute values of the RBM parameters, for example by using weight-decay (see the discussion by Fischer and Igel (2010a)). Further, the bound increases with the number of RBM variables and decreases with increasing k. The latter underpins that

larger values of k stabilize CD learning and that increasing k dynamically when the weights increase may be a good learning strategy (Bengio and Delalleau, 2009).

Chapter 6

A bound for the convergence rate of parallel tempering for sampling RBMs

---

This chapter is based on the manuscript "A bound for the convergence rate of parallel tempering for sampling restricted Boltzmann machines" by A. Fischer and C. Igel, submitted.

Abstract

Sampling from Restricted Boltzmann Machines (RBMs) is done by Markov Chain Monte Carlo (MCMC) methods. The faster the convergence of the Markov chain, the more efficiently can high quality samples be obtained. This is also important for robust training of RBMs, which usually relies on sampling. Parallel Tempering (PT), an MCMC method that maintains several replicas of the original chain at higher temperatures, has been successfully applied for RBM training. We present the first analysis of the convergence rate of PT for sampling from binary RBMs. We find an exponential dependency on the size of the RBM and the sum of the absolute values of the RBM parameters, similar to bounds on the approximation error for contrastive divergence learning.

6.1 Introduction

Restricted Boltzmann Machines (RBMs) are probabilistic graphical models corresponding to stochastic neural networks (Smolensky, 1986; Hinton, 2002) (see the article by Fischer and Igel (2014) for a recent review). They are applied in many machine learning tasks, notably they serve as building blocks of deep belief networks (Hinton and Salakhutdinov, 2006). Markov Chain Monte Carlo (MCMC) methods are used to sample from RBMs, and chains that quickly converge to their stationary distribution are desirable to efficiently get high quality samples. Adaptation of the RBM model parameters typically corresponds to gradient-based likelihood maximization given training data. As computing the exact gradient is usually computationally not tractable, sampling-based methods are employed to approximate the likelihood gradient. It has been shown that inaccurate approximations can deteriorate the learning process (e.g., Fischer and Igel, 2010a), and for the most popular learning scheme *Contrastive Divergence learning* (CD, Hinton, 2002) the approximation quality has been analyzed (Bengio and Delalleau, 2009; Fischer and Igel, 2011a). The quality of the approximation depends, among others, on how quickly the Markov chain approaches the stationary distribution, that is, on its mixing rate.

To improve RBM learning, *Parallel Tempering* (PT, Geyer, 1991) has successfully been used as a sampling method in RBM training (Salakhutdinov, 2009; Desjardins et al., 2010b; Cho et al., 2010; Fischer and Igel, 2014). Parallel tempering introduces supplementary Gibbs chains that sample from smoothed replicas of the original distribution—with the goal of improving the mixing rate. However, so far there exist no published attempts to analyze the mixing rate of PT applied to RBMs. Based on the work by Woodard et al. (2009b), we provide the first such analysis. After introducing the basic concepts, section 6.3 states our main result. Section 6.4 summarizes general theorems and the findings by Woodard et al. (2009b) required for our proof in section 6.5, which is followed by a discussion.

6.2 Background

In the following, we will give a brief introduction to RBMs and the relation between the mixing rate and the spectral gap of a Markov chain. Afterwards we will describe the parallel tempering algorithm and its application to sampling from RBMs.

6.2.1 Restricted Boltzmann machines

Restricted Boltzmann machines are probabilistic undirected graphical models (Markov random fields). Their structure is a bipartite graph connecting a set of m visible

random variables $\boldsymbol{V} = (V_1, V_2, \ldots, V_m)$ modeling observations to n hidden (latent) random variables $\boldsymbol{H} = (H_1, H_2, \ldots, H_n)$ capturing dependencies between the visible variables. In binary RBMs the state space of one single variable is given by $\Omega = \{0, 1\}$ and accordingly $(\boldsymbol{V}, \boldsymbol{H}) \in \{0, 1\}^{m+n}$. The joint probability distribution of $(\boldsymbol{V}, \boldsymbol{H})$ is given by the Gibbs distribution

$$\pi(\boldsymbol{v}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{Z} \ ,$$

with energy

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

and real-valued connection weights w_{ij} and bias parameters b_j and c_i for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. The normalization constant Z, also called partition function, is given by $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))$.

Training an RBM means adapting its parameters such that the distribution of $\boldsymbol{V}$ models a distribution underlying some observed data. In practice, this training corresponds to performing stochastic gradient descent on the log-likelihood of the weight and bias parameters given sample (training) data. The gradient of the log-likelihood given a single training sample $\tilde{\boldsymbol{v}}$ is given by

$$\frac{\partial \ln \pi(\tilde{\boldsymbol{v}} \,|\, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_{\boldsymbol{h}} \pi(\boldsymbol{h} \,|\, \tilde{\boldsymbol{v}}) \frac{\partial E(\tilde{\boldsymbol{v}}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v},\boldsymbol{h}} \pi(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \ , \tag{6.1}$$

where $\boldsymbol{\theta}$ is the vector collecting all parameters. Since the expectation under the model distribution in the second term on the right hand side can not be computed efficiently (it is exponential in $\min(n, m)$), it is approximated by MCMC methods in RBM training algorithms.

6.2.2 Mixing rates and the spectral gap

A homogeneous Markov chain on a discrete state space Ω can be described by a transition matrix $P = (p_{i,j})_{i,j\in\Omega}$, where $p_{i,j}$ is the probability to move from i to j in one step of the Markov chain. We also refer to this probability as $P(i, j)$, and accordingly $P^k(i, j)$ gives the probability to move from i to j in k steps of the chain.

Markov chain Monte Carlo methods make use of the fact that an ergodic Markov chain on Ω with transition matrix P and equilibrium distribution π satisfies $P^k(x, y) \to \pi(y)$ as $k \to \infty$ for all $x, y \in \Omega$. It is important to study the convergence rates of the Markov chain in order to find out how large k has to be to ensure that $P^k(x, y)$ is suitably close to $\pi(y)$. One way to measure the closeness to stationarity is the total variation distance $\|P^k(x, \cdot) - \pi\| = \frac{1}{2}\sum_y |P^k(x, y) - \pi(y)|$ for arbitrary starting states x. Reversibility of the chain implies that the eigenvalues of P are real-valued, and

we sort them by value $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq -1$ or by their absolute values $1 = \lambda_{|1|} \geq |\lambda_{|2|}| \geq \cdots \geq |\lambda_{|r|}| \geq 0$ with $r = |\Omega|$. The value $\mathrm{Gap}(P) = 1 - \lambda_2$ is called the *spectral gap* of P, and $|\lambda_{|2|}|$ is often referred to as *Second Largest Eigenvalue Modulus* (SLEM).

Many bounds on convergence rates are based on the SLEM. Diaconis and Saloff-Coste (1998), for example, prove

$$\|P^k(x,\cdot) - \pi\| \leq \frac{1}{2\sqrt{\pi(x)}} \lambda_{|2|}^k \ . \tag{6.2}$$

If P is positive definite all eigenvalues are non-negative. In this case $\lambda_2 = \lambda_{|2|}$ and we can replace $\lambda_{|2|}$ in (6.2) (and similar bounds) by $1 - \mathrm{Gap}(P)$. We can make an arbitrary transition matrix Q positive definite by skipping the move each time with probability $\frac{1}{2}$, that is, by considering the transition matrix $P = \frac{1}{2}I + \frac{1}{2}Q$. This makes P in the long run two times slower than Q.

6.2.3 Parallel tempering

Consider a multi modal target density π on a state space Ω from which one would like to sample via a Markov chain. The transition steps of the Metropolis-Hastings algorithm move only locally in space so that the resulting Markov chain may move between the modes of π only infrequently. The parallel tempering (PT) algorithm (Geyer, 1991) tries to overcome this by introducing supplementary Markov chains which are "flattened" or "smoothed" versions of π. These are the so called *tempered* densities π_t, $t = 0, \ldots, N$ fulfilling $\pi_t(\boldsymbol{z}) \propto \pi(\boldsymbol{z})^{\beta_t}$ for $\boldsymbol{z} \in \Omega$ with *inverse temperatures* β_t. The inverse temperatures satisfy $0 \leq \beta_0 < \cdots < \beta_N = 1$. Parallel tempering now constructs a Markov chain on the joint state space $\Omega_{\mathrm{PT}} = \Omega^{N+1}$ of the tempered distributions. The chain takes values $\boldsymbol{x} = (\boldsymbol{x}_{[0]}, \ldots, \boldsymbol{x}_{[N]}) \in \Omega_{\mathrm{PT}}$ and converges to the stationary distribution

$$\pi_{\mathrm{PT}}(\boldsymbol{x}) = \prod_{t=0}^{N} \pi_t(\boldsymbol{x}_{[t]}) \ .$$

The PT transition matrix consists out of two components, one for updating the states of the individual tempered chains and one allowing swaps between states of adjacent tempered chains. We will denote the corresponding transition matrices by T and Q, respectively. For the considerations in this paper, we add a probability of $\frac{1}{2}$ of staying in the current state each time T or Q is applied to guarantee positive definiteness and thus positive eigenvalues. We define the PT matrix as $P = QTQ$. A transition matrix constructed like this assures reversibility (unlike TQ) and is used in the analysis by Madras and Zheng (2003) and Woodard et al. (2009b).

The matrices T and Q are defined as follows: T chooses $t \in \{0, \ldots, N\}$ uniformly and updates the state of π_t based on some transition matrix T_t, which is reversible with

respect to π_t. This can, for example, be the Metropolis-Hastings or a Gibbs matrix (Brügge et al., 2013). Thus, the probability of moving from $\boldsymbol{x} = (\boldsymbol{x}_{[0]}, \ldots, \boldsymbol{x}_{[N]}) \in \Omega_{\mathrm{PT}}$ to $\boldsymbol{y} = (\boldsymbol{y}_{[0]}, \ldots, \boldsymbol{y}_{[N]}) \in \Omega_{\mathrm{PT}}$ under T is given by

$$T(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2(N+1)} \sum_{t=0}^{N} T_t(\boldsymbol{x}_{[t]}, \boldsymbol{y}_{[t]}) I_{\{\boldsymbol{x}_{[-t]}\}}(\boldsymbol{y}_{[-t]}) + \frac{1}{2} I_{\{\boldsymbol{x}\}}(\boldsymbol{y}) \ ,$$

where $x_{[-t]} = (\boldsymbol{x}_{[0]}, \ldots, \boldsymbol{x}_{[t-1]}, \boldsymbol{x}_{[t+1]}, \ldots, \boldsymbol{x}_{[N]}) \in \Omega^N$ denotes the vector of all except the t-th component and, for any set B, $I_B(y) = 1$ if $y \in B$ and $I_B(y) = 0$, otherwise.

The swapping matrix Q proposes to swap state $\boldsymbol{x}_{[t]}$ and $\boldsymbol{x}_{[t+1]}$ by sampling t uniformly from $\{0, \ldots, N-1\}$. The proposed swap is accepted with the Metropolis probability

$$a(\boldsymbol{x}, t) = \min \left\{ 1, \frac{\pi_t(\boldsymbol{x}_{[t+1]}) \pi_{t+1}(\boldsymbol{x}_{[t]})}{\pi_t(\boldsymbol{x}_{[t]}) \pi_{t+1}(\boldsymbol{x}_{[t+1]})} \right\} \ , \tag{6.3}$$

which guarantees detailed balance. So the transition probability from $\boldsymbol{x} = (\boldsymbol{x}_{[0]}, \ldots, \boldsymbol{x}_{[N]})$ to $\boldsymbol{y} = (\boldsymbol{x}_{[0]}, \ldots, \boldsymbol{x}_{[t+1]}, \boldsymbol{x}_{[t]}, \ldots \boldsymbol{x}_{[N]})$ is given by

$$Q(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2N} a(\boldsymbol{x}, t) \ ,$$

the probability to stay in $\boldsymbol{x}$ is

$$Q(\boldsymbol{x}, \boldsymbol{x}) = 1 - \sum_{t=0}^{N-1} \frac{1}{2N} a(\boldsymbol{x}, t) \ ,$$

and the transition probability between all other states is 0.

By construction both T and Q are reversible with respect to π_{PT} and strictly positive definite due to their $\frac{1}{2}$ holding probability. From the definition of P it follows that both properties also hold for P. Thus, P has only positive real-valued eigenvalues and the convergence of the PT chain to the stationary distribution π_{PT} can be bounded in terms of (6.2) by replacing $\lambda_{|2|}$ by a lower bound for $\mathrm{Gap}(P)$.

6.2.4 Training RBMs with PT

The PT algorithm is often the sampling procedure of choice for RBM learning algorithms to approximate the gradient of the log-likelihood given in equation (6.1)(Desjardins et al., 2010b; Cho et al., 2010). In this setting, the tempered distributions are defined as

$$\pi_t(\boldsymbol{v}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}) \beta_t)}{Z_t}$$

with $Z_t = \sum_{\boldsymbol{v}, \boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}) \beta_t)$ and β_0 is typically set to 0, which we will also assume in our analysis. The transition matrices T_t for the single tempered chains are set to a 'randomized' version of blockwise Gibbs sampling: we choose to update either $\boldsymbol{V}$ or $\boldsymbol{H}$

with equal probability and then update all neurons in this layer simultaneously. The random choice of the layer leads to reversibility of T_t.

In this paper, we want to find a bound for the PT sampling as defined above applied to RBMs. Note that in the PT algorithm most often used in practice the transition operator differs from the one analyzed in this paper. It consists out of an update step followed by a swapping step. During the update step, one step of blockwise Gibbs sampling is performed in all tempered chains in parallel. During the swapping step, the states at all temperatures may be swapped based on the Metropolis probability $a(t, \boldsymbol{x})$ as defined in (6.3), with $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{h})$. The swapping is often organized in two substeps, where even temperatures are considered in the first and odd temperatures in the second substep. While this approach seems to work in practice, the corresponding transition matrix need not be reversible, which makes it difficult to obtain theoretical results on its mixing behavior.

6.3 Main result

Let π denote the Gibbs distribution of RBMs and π_t the corresponding tempered Gibbs distribution with inverse temperature β_t. Since the tempered chains are independent the joint distribution over all chains is given by $\pi_{\mathrm{PT}}((\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)) = \prod_{t=0}^{N} \pi_t(\boldsymbol{x}_t)$. Then for the spectral gap of PT sampling for RBMs it holds:

Theorem 6.1. *For the PT transition matrix P for an RBM with m visible and n hidden variables*

$$\begin{aligned}\mathrm{Gap}(P) \geq \min \Bigg\{ &\frac{\exp(-2\Delta)}{2^{12+2n}(N+1)^4}, \frac{\exp(-4\Delta)}{2^{12+n}(N+1)^4}, \\ &\frac{\exp(-2\Delta)}{2^{12+2m}(N+1)^4}, \frac{\exp(-4\Delta)}{2^{12+m}(N+1)^4} \Bigg\},\end{aligned} \tag{6.4}$$

with Δ being the sum over the absolute values of parameters of the RBM:

$$\Delta = \sum_{i,j} |w_{ij}| + \sum_j |b_j| + \sum_i |c_i| \ .$$

The proof is given in section 6.5. By combining this result with equation (6.2), we arrive at an upper bound on the rate of convergence of the PT chain.

Corollary 6.1. *Let P be the PT transition matrix for an RBM with m visible and n hidden variables and let π_{PT} be the joint distribution over all tempered chains. Then for any starting point $\boldsymbol{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ of the Markov chain the distance in variation to π_{PT} is bounded by*

$$\|P^k(\boldsymbol{x}, \cdot) - \pi_{PT}\| \leq \frac{1}{2\sqrt{\pi_{PT}(\boldsymbol{x})}}(1-g)^k \tag{6.5}$$

with

$$g = \min\left\{\frac{\exp(-2\Delta)}{2^{12+2n}(N+1)^4}, \frac{\exp(-4\Delta)}{2^{12+n}(N+1)^4}, \frac{\exp(-2\Delta)}{2^{12+2m}(N+1)^4}, \frac{\exp(-4\Delta)}{2^{12+m}(N+1)^4}\right\}$$

and Δ *as defined above.*

Theorem 6.1 bounds the gap of the PT product chain. As the original chain (the chain with inverse temperature β_N) never converges slower than this product chain (see 6.7 for a proof), corollary 6.1 also leads to an upper bound for the convergence rate of the chain of interest. Bounding the product chain leads to the dependency on the number N of tempered chains. However, to our knowledge all approaches analyzing PT suffer from this drawback (e.g., Woodard et al., 2009b; Madras and Zheng, 2003; Bhatnagar and Randall, 2004).

6.4 Bounding the spectral gap of PT

This section summarizes definitions and results required for the proof of Theorem 6.1. We state important results for bounding the spectral gap of general Markov chains and of the general PT algorithm and recall the link between SLEM and *Dobrushin's coefficient*.

6.4.1 Definitions

For any transition matrix P reversible with respect to a distribution π and any subset A of the state space Ω of P, the *restriction* of P to A is defined as:

$$P|_A(x, B) = P(x, B) + I_B(x)P(x, A^c) \tag{6.6}$$

for $x \in A, B \subset A$. In this way transitions that would leave A are prohibited and the probabilities to stay in A are increased correspondingly.

Given a partition $\mathcal{A} = \{A_j : j = 1, \ldots, J\}$ of a finite Ω such that

$$\pi(A_j) = \sum_{x \in A_j} \pi(x) > 0$$

for all j, the *projection matrix* of P with respect to $\mathcal{A}$ is given by the transition operator $\bar{P}$ with

$$\bar{P}(i, j) = \frac{1}{\pi(A_i)} \sum_{x \in A_i} \sum_{y \in A_j} \pi(x)P(x, y) \tag{6.7}$$

for $i, j \in \{1, \ldots, J\}$.

6.4.2 General results

As Woodard et al. (2009b) show based on the results from Caracciolo et al. (1992) (which were first published in the work of Madras and Randall (2002)) and the results from Madras and Zheng (2003), one can formulate the following theorem

Theorem 6.2. *Let P be a transition matrix reversible with respect to a distribution π on a state space Ω. Let $\{A_j : j = 1, \dots, J\}$ be any partition of Ω such that $\pi(A_j) > 0$ for all j. Define $P_{|A_j}$ as in* (6.6) *and $\bar{P}$ as in* (6.7)*. If P is nonnegative definite, it holds*

$$\operatorname{Gap}(P) \geq \frac{1}{2} \operatorname{Gap}(\bar{P}) \min_{j=1,\dots,J} \operatorname{Gap}(P_{|A_j}) \ .$$

Combining the theorem for the comparison of the Dirichlet-forms of two Markov chains given by Diaconis and Saloff-Coste (1993) with Reyleigh's theorem (e.g., Brémaud, 1999, p. 205) we get

Theorem 6.3. *Let P and Q be transition matrices on a finite state space Ω, reversible with respect to densities π_P and π_Q respectively. Denote by $E_P = \{(x, y) : \pi_P(x)P(x, y) > 0\}$ and $E_Q = \{(x, y) : \pi_Q(x)Q(x, y) > 0\}$ the edge sets of the corresponding transition graphs. For each pair $x \neq y$ such that $(x, y) \in E_Q$ fix a path $\gamma_{x,y} = (x = x_0, x_1, \dots x_k = y)$ of length $|\gamma_{x,y}| = k$ such that $(x_i, x_{i+1}) \in E_P$ for $i \in \{0, \dots, k-1\}$ and define*

$$c = \max_{(z,w)\in E_P} \left\{ \frac{1}{\pi_P(z)P(z, w)} \sum_{\gamma_{x,y}:(z,w)\in\gamma_{x,y}} |\gamma_{x,y}| \pi_Q(x)Q(x, y) \right\} \ .$$

Then it holds $\operatorname{Gap}(Q) \leq c \operatorname{Gap}(P)$.

The following theorem deals with product chains (Diaconis and Saloff-Coste, 1996, Lemma 3.2):

Theorem 6.4. *For any natural number N and $t = 0, \dots, N$ let P_t be a π_t-reversible transition matrix on a state space Ω_t. Let P be the transition matrix on $\Omega = \prod_t \Omega_t$ given by*

$$P(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t=0}^{N} b_t P_t(x_{[t]}, y_{[t]}) I_{\{\boldsymbol{x}_{[-t]}\}}(\boldsymbol{y}_{[-t]}) \ , \boldsymbol{x}, \boldsymbol{y} \in \Omega$$

for some set set of $b_t > 0$ such that $\sum_t b_t = 1$ and where $x_{[t]}$ denotes the t-th entry of vector $\boldsymbol{x}$ and $\boldsymbol{x}_{[-t]}$ all except the t-th one. A Markov chain with transition matrix P is called a product chain. It is reversible with respect to $\pi(\boldsymbol{x}) = \prod_t \pi_t(x_{[t]})$ and

$$\operatorname{Gap}(P) = \min_{t=0,\dots,N} b_k \operatorname{Gap}(P_t) \ .$$

The next theorem gives an upper bound on the SLEM (e.g., see Brémaud, 1999, p. 237).

Theorem 6.5. *The second largest eigenvalue modulus* $|\lambda_{|2|}|$ *of a transition matrix* P *can be bounded from above by* Dobrushin's *coefficient:*

$$D(P) = \frac{1}{2} \max_{i,j \in \Omega} \sum_{k \in \Omega} |p_{ik} - p_{jk}| = 1 - \min_{i,j \in \Omega} \sum_{k \in \Omega} \min\{p_{ik}, p_{jk}\}$$

6.4.3 Bounding the spectral gap of PT

Now we consider a PT chain as defined in section 6.2.3 for some density π of interest on the state space Ω and corresponding tempered chains π_t, $t = 0, \ldots, N$, each associated with an inverse temperature parameter β_t, such that $0 \leq \beta_0 < \cdots < \beta_N = 1$, and a transition matrix T_t. Then a bound for the spectral gap of the PT transition matrix is given by the following theorem by Woodard et al. (2009b):

Theorem 6.6. *Given any partition* $A = \{A_j : j = 1, \ldots, J\}$ *of the state space* Ω *such that* $\pi_t(A_j) > 0$ *for all* j *and* t *then it holds*

$$\text{Gap}(P) \geq \frac{\gamma(A)^{J+3} \delta(A)^2}{2^{12}(N+1)^4 J^3} \text{Gap}(\bar{T}_0) \min_{t,j} \text{Gap}(T_t|_{A_j}) \tag{6.8}$$

with

$$\gamma(A) = \min_j \prod_{t=1}^{N} \min\left\{1, \frac{\pi_{t-1}(A_j)}{\pi_t(A_j)}\right\},$$
$$\delta(A) = \min_{t,j} \frac{\sum_{x \in A_j} \min\{\pi_t(x), \pi_{t+1}(x)\}}{\max\{\pi_t(A_j), \pi_{t+1}(A_j)\}}.$$

6.5 Proof of the main result

To apply the results from the previous section to RBMs, a suitable partitioning of the RBM state space is needed. For a binary RBM with m visible and n hidden neurons the state space is $\Omega = \{0,1\}^{n+m}$. Let $A = \{A_j : j = 1, \ldots, 2^n\}$ be the partition of Ω where each subset A_j contains all states having the same state of the hidden neurons, i.e., $A_j = \{(\boldsymbol{v}, \boldsymbol{h}) \in \Omega | \boldsymbol{h} = \boldsymbol{h}_j\}$ if $\boldsymbol{h}_j$ denotes the j-th state of the hidden random vector . Thus, we get 2^n subsets $A_1, \ldots, A_{2^n}$ and each A_j contains 2^m elements. Using this idea, we can prove Theorem 6.1 based on the proof of Theorem 6.6. Our proof can be divided into 6 steps. Steps 1, 2, and 4 are directly taken from Woodard et al. (2009b) and are accordingly formulated for general PT chains.

The basic ideas behind the steps of the proof can be outlined as follows. The gap of the PT transition matrix P depends on (a) how well the single tempered chains mix inside the single subsets $A_1, ...A_J$, (b) how well the chain at the lowest temperature mixes between the subsets, and (c) the mixing properties between the chains at the different temperatures. In step 1, Gap(P) is bounded in terms of the gaps of two transition matrices, one depending on (a) and the other depending on (b) and (c). The

first one is further bounded in steps 2 and 3 and the second one in 4 and 5. Step 6 puts everything together.

Step 1: Consider the state $\boldsymbol{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N) \in \Omega^{N+1}$ of the PT chain. Let the signature be the vector $s(\boldsymbol{x}) = (\sigma_0, \ldots, \sigma_N)$ with $\sigma_t = j$ if $\boldsymbol{x}_t \in A_j$ for $t = 0, \ldots, N$. Since the partition of Ω consists out of J subsets A_j and we have $N+1$ temperatures, a signature lives in $\Sigma = \{1 \ldots, J\}^{N+1}$.

For a fixed $\boldsymbol{\sigma} \in \Sigma$ let us now define $\Omega_{\boldsymbol{\sigma}} = \{\boldsymbol{x} \in \Omega^{N+1} : s(\boldsymbol{x}) = \boldsymbol{\sigma}\}$. Then all possible $\boldsymbol{\sigma} \in \Sigma$ induce a partition $\{\Omega_{\boldsymbol{\sigma}}\}_{\boldsymbol{\sigma} \in \Sigma}$ of the PT-state space $\Omega_{\mathrm{PT}} = \Omega^{N+1}$.

Let $P_\sigma = P_{|\Omega_{\boldsymbol{\sigma}}}$ now be the restriction of P to $\Omega_{\boldsymbol{\sigma}}$ as defined in equation (6.6). And let $\bar{P}$ denote the projection matrix of P with respect to $\{\Omega_{\boldsymbol{\sigma}}\}_{\boldsymbol{\sigma} \in \Sigma}$ as defined in (6.7). Now we can apply Theorem 6.2 and get

$$\mathrm{Gap}(P) \geq \frac{1}{2} \mathrm{Gap}(\bar{P}) \min_{\sigma \in \Sigma} \mathrm{Gap}(P_\sigma) \ . \tag{6.9}$$

Step 2: Woodard et al. now proceed by bounding $\mathrm{Gap}(P_\sigma)$ and $\mathrm{Gap}(\bar{P})$ separately. For $\mathrm{Gap}(P_\sigma)$ they show

$$\mathrm{Gap}(P_\sigma) \geq \frac{1}{8(N+1)} \min_{t,j} \mathrm{Gap}(T_t|_{A_j}) \ . \tag{6.10}$$

Step 3: We will now drive a lower bound for $\min_{t,j} \mathrm{Gap}(T_t|_{A_j})$. The standard transition matrix T_t used in our (tempered) RBMs corresponds to randomized blockwise Gibbs sampling as described above. Thus, the transition probability from $(\boldsymbol{v}, \boldsymbol{h})$ to $(\boldsymbol{v}', \boldsymbol{h})$ is given by

$$T_t((\boldsymbol{v}, \boldsymbol{h})), (\boldsymbol{v}', \boldsymbol{h})) = \frac{1}{2} \pi_t(\boldsymbol{v}' | \boldsymbol{h}) \ ,$$

and the probability to change the state of the hidden variables is accordingly

$$T_t((\boldsymbol{v}, \boldsymbol{h})), (\boldsymbol{v}, \boldsymbol{h}')) = \frac{1}{2} \pi_t(\boldsymbol{h}' | \boldsymbol{v}) \ .$$

Based on (6.6) for the restriction of T_t to A_j it holds:

$$T_t|_{A_j}((\boldsymbol{v}, \boldsymbol{h}_j), (\boldsymbol{v}', \boldsymbol{h}_j)) = T_t((\boldsymbol{v}, \boldsymbol{h}_j), (\boldsymbol{v}', \boldsymbol{h}_j)) + I_{\{(\boldsymbol{v}, \boldsymbol{h}_j)\}}((\boldsymbol{v}', \boldsymbol{h}_j)) T_t((\boldsymbol{v}, \boldsymbol{h}_j), A_j^c)$$

for $(\boldsymbol{v}, \boldsymbol{h}_j), (\boldsymbol{v}', \boldsymbol{h}_j) \in A_j$. Thus, for $\boldsymbol{v} \neq \boldsymbol{v}'$

$$T_t|_{A_j}((\boldsymbol{v}, \boldsymbol{h}_j), (\boldsymbol{v}', \boldsymbol{h}_j)) = \frac{1}{2} \pi_t(\boldsymbol{v}' | \boldsymbol{h}_j) \ .$$

and the probability to stay in the same state is

$$T_t|_{A_j}((\boldsymbol{v}, \boldsymbol{h}_j), (\boldsymbol{v}, \boldsymbol{h}_j)) = \frac{1}{2} \pi_t(\boldsymbol{v} | \boldsymbol{h}_j) + \sum_{\boldsymbol{h}} \frac{1}{2} \pi_t(\boldsymbol{h} | \boldsymbol{v}) = \frac{1}{2} \pi_t(\boldsymbol{v} | \boldsymbol{h}_j) + \frac{1}{2} \ .$$

Let us denote the SLEM of $T_t|_{A_j}$ by $\lambda_{|2|}^{T_t}$ and the second largest eigenvalue by $\lambda_2^{T_t}$. Based on Theorem 6.5 it holds

$$\mathrm{Gap}(T_t|A_j) = 1 - \lambda_2^{T_t} \geq 1 - \lambda_{|2|}^{T_t} \geq 1 - D(T_t|_{A_j}) \ . \tag{6.11}$$

To upper bound the Drobushin's coefficient of $T_t|_{A_j}$ first note that for all $\boldsymbol{v}$, $\boldsymbol{v}'$:

$$\begin{aligned}
&\sum_{\hat{\boldsymbol{v}}} \min\{T_t|_{A_j}((\boldsymbol{v},\boldsymbol{h}_j),(\hat{\boldsymbol{v}},\boldsymbol{h}_j)), T_t|_{A_j}((\boldsymbol{v}',\boldsymbol{h}_j),(\hat{\boldsymbol{v}},\boldsymbol{h}_j))\} \\
&\qquad = \sum_{\hat{\boldsymbol{v}}:\hat{\boldsymbol{v}}\neq\boldsymbol{v}\wedge\hat{\boldsymbol{v}}\neq\boldsymbol{v}'} \frac{1}{2}\pi_t(\hat{\boldsymbol{v}}|\boldsymbol{h}_j) + \min\left\{\frac{1}{2}\pi_t(\boldsymbol{v}|\boldsymbol{h}_j), \frac{1}{2}\pi_t(\boldsymbol{v}|\boldsymbol{h}_j) + \frac{1}{2}\right\} \\
&\qquad\qquad + \min\left\{\frac{1}{2}\pi_t(\boldsymbol{v}'|\boldsymbol{h}_j), \frac{1}{2}\pi_t(\boldsymbol{v}'|\boldsymbol{h}_j) + \frac{1}{2}\right\} \\
&\qquad\qquad\qquad = \sum_{\hat{\boldsymbol{v}}'} \frac{1}{2}\pi_t(\hat{\boldsymbol{v}}|\boldsymbol{h}_j) = \frac{1}{2} \ .
\end{aligned}$$

Now it is easy to see that $D(T_t|_{A_j}) = 1 - \frac{1}{2} = \frac{1}{2}$ and insertion into (6.11) gives $\mathrm{Gap}(T_t|_{A_j}) \geq 1 - D(T_t|_{A_j}) \geq \frac{1}{2}$. Thus, we finally get by insertion into equation (6.10)

$$\mathrm{Gap}(P_\sigma) \geq \frac{1}{16(N+1)} \ . \tag{6.12}$$

Step 4: For bounding $\mathrm{Gap}(\bar{P})$ first note that $\bar{P}$ is reversible with respect to the probability mass function

$$\pi^*(\boldsymbol{\sigma}) = \pi_{\mathrm{PT}}(\Omega_{\boldsymbol{\sigma}}) = \prod_{t=0}^{N} \pi_t(A_{\sigma_t}) \ , \forall \sigma \in \Sigma$$

and that for any $\boldsymbol{\sigma}$, $\boldsymbol{\tau} \in \Sigma$, the probability of moving from $\Omega_{\boldsymbol{\sigma}}$ to $\Omega_{\boldsymbol{\tau}}$ under P at stationarity is given by

$$\bar{P}(\boldsymbol{\sigma},\boldsymbol{\tau}) = \frac{1}{\pi_{\mathrm{PT}}(\Omega_{\boldsymbol{\sigma}})} \sum_{\boldsymbol{x}\in\Omega_{\boldsymbol{\sigma}}} \sum_{\boldsymbol{y}\in\Omega_{\boldsymbol{\tau}}} \pi_{pt}(\boldsymbol{x}) P(\boldsymbol{x},\boldsymbol{y}) \ .$$

Woodard et al. show that a first bound can be given in terms of a transition matrix T^* constructed as follows: with probability $\frac{1}{2}$ perform a transition according to $\bar{Q}$, with probability $\frac{1}{2(N+1)}$ draw σ_0 according to the distribution $\pi_0^*(\sigma_0) = \pi_0(A_{\sigma_0})$ for $\sigma_0 = 1, \ldots, J$, otherwise hold. The bound is then given by

$$\mathrm{Gap}(\bar{P}) \geq \frac{\mathrm{Gap}(T^*)\,\mathrm{Gap}(\bar{T}_0)}{4} \ . \tag{6.13}$$

Step 5: Woodard et al. proceed by further bounding $\mathrm{Gap}(T^*)$ based on Theorem 6.3 by comparing T^* to another π^*-reversible transition matrix T^{**}. The transition matrix T^{**} chooses t uniformly from $\{0, \ldots, N\}$ and then draws σ_t according to the distribution $\pi_t^*(\sigma_t) = \pi_t(A_{\sigma_t})$.

To ease the notation let us now denote $\boldsymbol{\sigma}_{[i,j]} = (\sigma_0, \ldots, \sigma_{i-1}, j, \sigma_{i+1}, \ldots, \sigma_N)$ and $j^* = \arg\max_{j \in \{1,\ldots,J\}} \pi_N(A_j)$. For the application of Theorem 6.3 for each edge $(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]})$ in the transition graph of T^{**} let us define a path $\gamma_{\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]}}$ in the transition graph of T^* as follows:

1. change σ_0 to j^*;
2. swap j^* "up" to level i;
3. swap new σ_{i-1} (formerly σ_i) "down" to level 0;
4. change value at level 0 to j (from former σ_i);
5. swap j "up" to level i;
6. swap j^* (now at level $i-1$) "down" to level 0;
7. change value at level 0 to σ_0 (from j^*).

We derive an upper bound of the constant c of Theorem 6.3 by splitting it into two terms and bounding each term separately. Here c is the maximum with respect to $\boldsymbol{\tau}$ and $\boldsymbol{\xi}$ (with $\pi^*(\boldsymbol{\tau})T^*(\boldsymbol{\tau}, \boldsymbol{\xi}) > 0$) of

$$\frac{1}{\pi^*(\boldsymbol{\tau})T^*(\boldsymbol{\tau}, \boldsymbol{\xi})} \sum_{\gamma_{\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]}} : (\boldsymbol{\tau},\boldsymbol{\xi}) \in \gamma_{\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]}}} |\gamma_{x,y}| \pi^*(\boldsymbol{\sigma}) T^{**}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]}) \ .$$

Woodard et al. show that for the above-defined paths

$$\sum_{\gamma_{\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]}} : (\boldsymbol{\tau},\boldsymbol{\xi}) \in \gamma_{\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]}}} |\gamma_{\sigma,\sigma_{[i,j]}}| \leq 16(N+1)^2 J^2 \tag{6.14}$$

for any edge $(\boldsymbol{\tau}, \boldsymbol{\xi})$ in the graph of T^*.

Now we upper bound $\frac{\pi^*(\boldsymbol{\sigma})T^{**}(\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]})}{\pi^*(\boldsymbol{\tau})T^*(\boldsymbol{\tau},\boldsymbol{\xi})}$ by splitting it into $\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})}$ and $\frac{T^{**}(\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]})}{T^*(\boldsymbol{\tau},\boldsymbol{\xi})}$ and bounding both terms separately. For bounding $\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})}$ first note that any state in the stages 1 or 2 of the path from $\boldsymbol{\sigma}$ to $\boldsymbol{\sigma}_{[i,j]}$ as given above is of the form $\boldsymbol{\tau} = (\sigma_1, \ldots, \sigma_l, j^*, \sigma_{l+1}, \ldots, \sigma_N)$ for some $l \in \{0, \ldots, i\}$. Therefore,

$$\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} = \left[\prod_{k=1}^{l} \frac{\pi_k(A_{\sigma_k})}{\pi_{k-1}(A_{\sigma_k})}\right] \frac{\pi_0(A_{\sigma_0})}{\pi_l(A_{j^*})} \ . \tag{6.15}$$

For the Boltzmann distribution of RBMs we have

$$\frac{\pi_0(A_{\sigma_0})}{\pi_l(A_{j^*})} = \frac{Z_l}{Z_0} \frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_1})\beta_0)}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_l)} \leq \frac{Z_l}{Z_0} \frac{2^m \exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_0)}{2^m \exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_l)}$$

and for the first term on the left side of equation (6.15)

$$\prod_{k=1}^{l} \frac{\pi_k(A_{\sigma_k})}{\pi_{k-1}(A_{\sigma_k})} = \frac{Z_0}{Z_l} \prod_{k=1}^{l} \frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_k)}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_{k-1})} \ .$$

Now consider that

$$\frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_k)}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_{k-1})} \leq \frac{2^m \exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_k)}{2^m \exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{k-1})} \tag{6.16}$$

because we can write

$$\frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_k)}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_{k-1})} = \frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_k + \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})\beta_{k-1} + \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \ ,$$

which makes all arguments of the exponential function in the terms of denominator and numerator nonnegative. The function $\frac{R_k + \exp(-x\beta_k + y)}{R_{k-1} + \exp(-x\beta_{k-1} + y)}$ is monotone decreasing in x for $x \leq y$ for $1 \geq \beta_k \geq \beta_{k-1} \geq 0$, and for each value of $\boldsymbol{v}$ we can write $x = E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_k})$ and $y = \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})$ and fix R_k and R_{k-1} to the remaining terms in numerator and denominator, respectively. Thus, the expression gets maximal if we replace x by $\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})$. This can be done for all values of $\boldsymbol{v}$. So we can write:

$$\begin{aligned}
\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} &\leq \frac{Z_0}{Z_l} \left[\prod_{k=1}^{l} \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_k)}{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{k-1})} \right] \frac{Z_l}{Z_0} \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_0)}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_l)} \\
&= \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_l)}{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_0)} \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_0)}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_l)} \\
&= \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_l)}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_l)} \leq \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*}))} \ .
\end{aligned}$$

Any state $\boldsymbol{\tau}$ in stage 3 of the path is of the form

$$\boldsymbol{\tau} = (\sigma_1, \ldots, \sigma_l, \sigma_i, \sigma_{l+1}, \ldots, \sigma_{i-1}, j^*, \sigma_{i+1}, \ldots, \sigma_N)$$

for some $l \in \{0, \ldots, i-1\}$. Thus,

$$\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} = \left[\prod_{k=1}^{l} \frac{\pi_k(A_{\sigma_k})}{\pi_{k-1}(A_{\sigma_k})} \right] \frac{\pi_0(A_{\sigma_0})}{\pi_i(A_{j^*})} \frac{\pi_i(A_{\sigma_i})}{\pi_l(A_{\sigma_i})} \ .$$

In an analogous way as above we get

$$\begin{aligned}
\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} &\leq \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_l)}{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_0)} \times \\
&\qquad \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_0)}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_i)} \frac{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_i)}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_l)} \\
&= \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_l)}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_l)} \frac{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_i)}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_i)} \\
&\leq \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{N-1})}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{N-1})} \frac{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*}))}
\end{aligned}$$

Any state $\boldsymbol{\tau}$ in stage 4 is given by $\boldsymbol{\tau} = (j, \sigma_1, \ldots, \sigma_{i-1}, j^*, \sigma_{i+1}, \ldots, \sigma_N)$. Therefore,

$$\begin{aligned}\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} &= \frac{\pi_0(A_{\sigma_0})}{\pi_0(A_j)} \frac{\pi_i(A_{\sigma_i})}{\pi_i(A_{j^*})} \\ &= \frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_0})\beta_0)}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_j)\beta_0)} \frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_i})\beta_i)}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{j*})\beta_i)} \\ &\overset{\beta_0=0}{=} \frac{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{\sigma_i})\beta_i)}{\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_{j*})\beta_i)} \leq \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j*}))} \quad .\end{aligned}$$

Any state in stage 5 or 6 can be written as

$$\boldsymbol{\tau} = (\sigma_1, \ldots, \sigma_l, j, \sigma_{l+1}, \ldots, \sigma_{i-1}, j^*, \sigma_{i-1}, \sigma_N)$$

or

$$\boldsymbol{\tau} = (\sigma_1, \ldots, \sigma_l, j^*, \sigma_{l+1}, \ldots, \sigma_{i-1}, j, \sigma_{i-1}, \sigma_N)$$

for some $l \in \{0, \ldots, i\}$. Therefore, it looks like any state in stage 3 where we replace either σ_i by j or σ_i by j^* and j^* by j, respectively. And thus for $\boldsymbol{\tau}$ in stage 5

$$\begin{aligned}\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} &= \left[\prod_{k=1}^{l} \frac{\pi_k(A_{\sigma_k})}{\pi_{k-1}(A_{\sigma_k})}\right] \frac{\pi_0(A_{\sigma_0})}{\pi_i(A_{j^*})} \frac{\pi_i(A_{\sigma_i})}{\pi_l(A_j)} \\ &\leq \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{N-1})}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{N-1})} \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*}))}\end{aligned}$$

and for $\boldsymbol{\tau}$ in stage 6:

$$\begin{aligned}\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} &= \left[\prod_{k=1}^{l} \frac{\pi_k(A_{\sigma_k})}{\pi_{k-1}(A_{\sigma_k})}\right] \frac{\pi_0(A_{\sigma_0})}{\pi_i(A_j)} \frac{\pi_i(A_{\sigma_i})}{\pi_l(A_{j^*})} \\ &\leq \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{N-1})}{\exp(-\max_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{h}_{j^*})\beta_{N-1})} \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}\end{aligned}$$

And finally any state in stage 7 is given by $\boldsymbol{\tau} = (\sigma_0, \sigma_1, \ldots, \sigma_{i-1}, j, \sigma_{i+1}, \ldots, \sigma_N)$ and thus

$$\frac{\pi^*(\boldsymbol{\sigma})}{\pi^*(\boldsymbol{\tau})} = \frac{\pi_i(A_{\sigma_i})}{\pi_i(A_j)} \leq \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \quad .$$

Thus, for any state $\boldsymbol{\tau}$ in the path $\gamma_{[\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]}]}$ we can upper bound $\pi^*(\boldsymbol{\sigma})/\pi^*(\boldsymbol{\tau})$ by

$$\frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{N-1})}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})\beta_{N-1})} \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \tag{6.17}$$

and for the states in stages 1, 4 and 7 we get a (tighter) upper bound by

$$\frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \quad . \tag{6.18}$$

Now we will bound the remaining term $\frac{T^{**}(\boldsymbol{\sigma},\boldsymbol{\sigma}_{[i,j]})}{T^*(\boldsymbol{\tau},\boldsymbol{\xi})}$. First note that

$$T^{**}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]}) = \frac{1}{N+1}\pi_i(A_j) \leq \frac{1}{N+1} \frac{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{2^n \exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \quad . \tag{6.19}$$

Any edge $(\boldsymbol{\tau}, \boldsymbol{\xi})$ on the path $\gamma_{[\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]}]}$ is either one, where we get $\boldsymbol{\xi}$ from $\boldsymbol{\tau} = (\tau_0, \ldots, \tau_N)$ by replacing τ_0 by some other state sampled from the distribution $\pi^*(\sigma_0) = \pi_0(A_{\sigma_0})$, which for $\beta_0 = 0$ is the uniform distribution over $\{A_0, \ldots, A_{2^n}\}$ (this happens with overall probability $\frac{1}{2(N+1)} \frac{1}{2^n}$ under T^*), or one, which we obtain by swapping two elements τ_t and τ_{t+1} according to $\bar{Q}$ (this happens with probability $\frac{1}{2}\bar{Q}(\boldsymbol{\tau}, \boldsymbol{\xi})$ under T^*).

In the first case the edge corresponds to stage 1, 4, or 7 of the path and

$$\frac{T^{**}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]})}{T^*(\boldsymbol{\tau}, \boldsymbol{\xi})} = \frac{2(N+1)\pi_i(A_j)}{(N+1)\pi_0(A_m)} = \frac{2\pi_i(A_j)}{1/2^n} \quad ,$$

with $m \in \{\sigma_0, j, j^*\}$. Using (6.19) this is bounded from above by

$$\frac{2^{n+1} \exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{2^n \exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \quad .$$

By joining this result with the upper bound given in equation (6.18) we get

$$\frac{\pi^*(\boldsymbol{\sigma}) T^{**}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]})}{\pi^*(\boldsymbol{\tau}) T^*(\boldsymbol{\tau}, \boldsymbol{\xi})} \leq \frac{2 \exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \quad .$$

In the second case the edge corresponds to stage 2, 3, 5, or 6. Recall that the probability to propose a certain swap according to Q is $\frac{1}{2N}$. The probability at stationarity of accepting the proposed swap between any two states $\boldsymbol{x}_{[t]} = (\boldsymbol{v}, \boldsymbol{h}_j) \in A_j$ and $\boldsymbol{x}_{[t+1]} = (\hat{\boldsymbol{v}}, \boldsymbol{h}_i) \in A_i$ under Q for any $t \in \{0, \ldots, N-1\}$ and any $i, j \in \{0, \ldots, 2^n\}$ is

$$\frac{1}{\pi_t(A_i)\pi_{t+1}(A_j)} \sum_{\boldsymbol{x}_{[t]} \in A_j} \sum_{\boldsymbol{x}_{[t+1]} \in A_i} \min\{\pi_t(\boldsymbol{x}_{[t]})\pi_{t+1}(\boldsymbol{x}_{[t+1]}), \pi_{t+1}(\boldsymbol{x}_{[t]})\pi_t(\boldsymbol{x}_{[t+1]})\} \quad . \tag{6.20}$$

These probabilities exactly describe the entries in $\bar{Q}$ and thus by bounding equation (6.20) we get a bound for $\bar{Q}(\boldsymbol{\tau}, \boldsymbol{\xi})$. We have

$$\begin{aligned}
&\sum_{\boldsymbol{x}_{[t]} \in A_j} \sum_{\boldsymbol{x}_{[t+1]} \in A_i} \min\{\pi_t(\boldsymbol{x}_{[t]})\pi_{t+1}(\boldsymbol{x}_{[t+1]}), \pi_{t+1}(\boldsymbol{x}_{[t]})\pi_t(\boldsymbol{x}_{[t+1]})\} \\
&\quad = \sum_{\boldsymbol{x}_{[t]} \in A_j} \sum_{\boldsymbol{x}_{[t+1]} \in A_i} \pi_t(\boldsymbol{x}_{[t]})\pi_{t+1}(\boldsymbol{x}_{[t+1]}) \min\left\{1, \frac{\pi_{t+1}(\boldsymbol{x}_{[t]})\pi_t(\boldsymbol{x}_{[t+1]})}{\pi_t(\boldsymbol{x}_{[t]})\pi_{t+1}(\boldsymbol{x}_{[t+1]})}\right\} \\
&\quad = \frac{1}{Z_t Z_{t+1}} \sum_{\boldsymbol{v}} \sum_{\hat{\boldsymbol{v}}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_j)\beta_t) \exp(-E(\hat{\boldsymbol{v}}, \boldsymbol{h}_i)\beta_{t+1}) \\
&\qquad\qquad \min\left\{1, \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}_j)\beta_{t+1}) \exp(-E(\hat{\boldsymbol{v}}, \boldsymbol{h}_i)\beta_t)}{\exp(-E(\boldsymbol{v}, \boldsymbol{h}_j)\beta_t) \exp(-E(\hat{\boldsymbol{v}}, \boldsymbol{h}_i)\beta_{t+1})}\right\}
\end{aligned}$$

$$= \frac{1}{Z_t Z_{t+1}} \sum_{\boldsymbol{v}} \sum_{\hat{\boldsymbol{v}}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_j)\beta_t) \exp(-E(\hat{\boldsymbol{v}}, \boldsymbol{h}_i)\beta_{t+1})$$
$$\min\{1, \exp\big((E(\hat{\boldsymbol{v}}, \boldsymbol{h}_i) - E(\boldsymbol{v}, \boldsymbol{h}_j))(\beta_{t+1} - \beta_t)\big)\}$$
$$\geq \frac{1}{Z_t Z_{t+1}} \exp\big((\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}) - \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))(\beta_{t+1} - \beta_t)\big)$$
$$\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_j)\beta_t) \sum_{\hat{\boldsymbol{v}}} \exp(-E(\hat{\boldsymbol{v}}, \boldsymbol{h}_i)\beta_{t+1})$$

and

$$\frac{1}{\pi_t(A_i)\pi_{t+1}(A_j)} = \frac{Z_t Z_{t+1}}{(\sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}_j)\beta_t))(\sum_{\hat{\boldsymbol{v}}} \exp(-E(\hat{\boldsymbol{v}}, \boldsymbol{h}_i)\beta_{t+1}))}$$

and thus a bound of (6.20) is given by $\frac{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}))}{\exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}))}$ and so

$$\bar{Q}(\boldsymbol{\tau}, \boldsymbol{\xi}) \geq \frac{\exp(-\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{2N \exp(-\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \ .$$

From these considerations follows

$$\frac{T^{**}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]})}{T^{*}(\boldsymbol{\tau}, \boldsymbol{\xi})} \leq \frac{2 \exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{2^n \exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}$$

and with (6.17)

$$\frac{\pi^{*}(\boldsymbol{\sigma}) T^{**}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{[i,j]})}{\pi^{*}(\boldsymbol{\tau}) T^{*}(\boldsymbol{\tau}, \boldsymbol{\xi})} \leq \frac{2 \exp(-4 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{2^n \exp(-4 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \ .$$

Putting these results and (6.14) together, an upper bound for the constant c from Theorem 6.3 is given by

$$c \leq \frac{32(N+1)^2 2^{2n} \exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \max\left\{1, \frac{\exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h})) 2^n}\right\}$$
$$= \frac{32(N+1)^2 2^{n} \exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \max\left\{2^n, \frac{\exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}\right\} \ .$$

Thus, applying Theorem 6.3 and Theorem 6.4, from which follows that $\mathrm{Gap}(T^{**}) = (N+1)^{-1}$ because all component chains of the product chain T^{**} have a spectral gap of 1, we get:

$$\mathrm{Gap}(T^{*}) \geq \frac{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{2^5 (N+1)^3 2^n \exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \times$$
$$\min\left\{\frac{1}{2^n}, \frac{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}\right\} \ . \quad (6.21)$$

Step 6: Insertion of (6.21) into (6.13) leads to

$$\mathrm{Gap}(\bar{P}) \geq \frac{\mathrm{Gap}(T^{*}) \, \mathrm{Gap}(\bar{T}_0)}{4}$$
$$\geq \frac{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{2^7 (N+1)^3 2^n \exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))} \times \min\left\{\frac{1}{2^n}, \frac{\exp(-2 \max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}{\exp(-2 \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}))}\right\}$$

using that $\mathrm{Gap}(\bar{T}_0) = 1$ because the transition probabilities are equal to the uniform distribution over the A_i independent of the current state (i.e., all entries in $\bar{T}_0$ equal $\frac{1}{2^n}$). Using this and (6.12) in (6.9) leads to

$$\begin{aligned}
\mathrm{Gap}(P) &\geq \frac{1}{2}\,\mathrm{Gap}(\bar{P}) \min_{\sigma\in\Sigma} \mathrm{Gap}(P_\sigma) \\
&\geq \frac{\exp(-2\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}))}{2^{12}(N+1)^4 2^n \exp(-2\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}))} \times \min\left\{\frac{1}{2^n}, \frac{\exp(-2\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}))}{\exp(-2\min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}))}\right\} \\
&= \min\left\{\frac{\exp(-2(\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}) - \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h})))}{2^{12}(N+1)^4 2^{2n}}, \right. \\
&\qquad \left. \frac{\exp(-4(\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}) - \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h})))}{2^{12}(N+1)^4 2^n}\right\} ,
\end{aligned}$$

which we further bound using

$$\max_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h}) - \min_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v},\boldsymbol{h})) \leq \Delta \ ,$$

with $\Delta = \sum_{i,j} |w_{ij}| + \sum_j |b_j| + \sum_i |c_i|$ summing the absolute values of parameters of the RBM. It is possible to repeat the proof while changing the roles of hidden and visible variables, which finally leads to (6.4).

6.6 Conclusion

We presented a first analysis of the convergence of the Markov chains in Parallel Tempering (PT) for sampling RBMs by deriving—an arguably loose, but non-trivial—bound on the spectral gap. We find a exponential dependency on the size of the two layers and the sum of the absolute values of the RBM parameters. The fewer the number of nodes and/or the smaller the parameters, the faster the convergence. This intuitive result resembles the bounds on the approximation bias in contrastive divergence learning (Fischer and Igel, 2011a). The observed difficulty to get rid of the exponential dependencies on the RBM complexity supports our hypothesis that RBM PT chains are not rapidly mixing. Because our analysis considers the convergence to the stationary distribution of the product chain consisting of all replicas, we get an—in this type of analysis inevitable—undesired additional linear dependency on the number of replicas.

6.7 Appendix

Relation between the convergence rates of the PT product chain and the original chain

In the following, we proof that bounding the convergence rate of the PT product chain also bounds the convergence rate of the the original chain (i.e., the chain with inverse temperature $\beta_N = 1$). Assume that for the product chain holds

$$\frac{1}{2}\sum_{\boldsymbol{x}} |P^k(\boldsymbol{y},\boldsymbol{x}) - \prod_{t=0}^{N}\pi_t(\boldsymbol{x}_{[t]})| < \epsilon \ ,$$

for an arbitrary starting point $\boldsymbol{y}$. Then we can write

$$\frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\sum_{\boldsymbol{x}_{[-N]}} |P^k(\boldsymbol{y},\boldsymbol{x}) - \prod_{t=0}^{N}\pi_t(\boldsymbol{x}_{[t]})| < \epsilon \ ,$$

which implies

$$\epsilon > \frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\sum_{\boldsymbol{x}_{[-N]}} \Big|P^k(\boldsymbol{y},\boldsymbol{x}) - \prod_{t=0}^{N}\pi_t(\boldsymbol{x}_{[t]})\Big| \geq \frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\Big|\sum_{\boldsymbol{x}_{[-N]}}(P^k(\boldsymbol{y},\boldsymbol{x}) - \prod_{t=0}^{N}\pi_t(\boldsymbol{x}_{[t]}))\Big| \ .$$

Thus, ϵ also upper bounds the variation distance for the original chain at temperature 1:

$$\begin{aligned}
\frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\Big|\sum_{\boldsymbol{x}_{[-N]}}(P^k(\boldsymbol{y},\boldsymbol{x}) - \prod_{t=0}^{N}\pi_t(\boldsymbol{x}_{[t]}))\Big| &= \frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\Big|\sum_{\boldsymbol{x}_{[-N]}}P^k(\boldsymbol{y},\boldsymbol{x}) - \sum_{\boldsymbol{x}_{[-N]}}\prod_{t=0}^{N}\pi_t(\boldsymbol{x}_{[t]})\Big| \\
&= \frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\Big|P_N^k(\boldsymbol{y},\boldsymbol{x}_{[0]}) - \sum_{\boldsymbol{x}_{[-N]}}\prod_{t=0}^{N}\pi_t(\boldsymbol{x}_{[t]})\Big| \\
&= \frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\Big|P_N^k(\boldsymbol{y},\boldsymbol{x}_{[0]}) - \pi_N(\boldsymbol{x}_{[0]})\sum_{\boldsymbol{x}_{[-N]}}\prod_{t=0}^{N-1}\pi_t(\boldsymbol{x}_{[t]})\Big| \\
&= \frac{1}{2}\sum_{\boldsymbol{x}_{[N]}}\Big|P_N^k(\boldsymbol{y},\boldsymbol{x}_{[0]}) - \pi_N(\boldsymbol{x}_{[0]})\Big| < \epsilon
\end{aligned}$$

Chapter 7

The flip-the-state transition operator for RBMs

This chapter is based on the manuscript "The flip-the-state transition operator for restricted Boltzmann machines" by K. Brügge, A. Fischer, and C. Igel published in *Machine Learning* 13, pp. 53-69, 2013.

Abstract

Most learning and sampling algorithms for Restricted Boltzmann Machines (RMBs) rely on Markov Chain Monte Carlo (MCMC) methods using Gibbs sampling. The most prominent examples are Contrastive Divergence learning (CD) and its variants as well as Parallel Tempering (PT). The performance of these methods strongly depends on the mixing properties of the Gibbs chain. We propose a Metropolis-type MCMC algorithm relying on a transition operator maximizing the probability of state changes. It is shown that the operator induces an irreducible, aperiodic, and hence properly converging Markov chain, also for the typically used periodic update schemes. The transition operator can replace Gibbs sampling in RBM learning algorithms without producing computational overhead. It is shown empirically that this leads to faster mixing and in turn to more accurate learning.

7.1 Introduction

Restricted Boltzmann Machines (RBMs, Smolensky, 1986; Hinton, 2002) are undirected graphical models describing stochastic neural networks. They have raised much attention recently as building blocks of deep belief networks (Hinton and Salakhutdinov, 2006). Learning an RBM corresponds to maximizing the likelihood of the parameters given data. Training large RBMs by steepest ascent on the log-likelihood gradient is in general computationally intractable, because the gradient involves averages over an exponential number of terms. Therefore, the computationally demanding part of the gradient is approximated by Markov Chain Monte Carlo (MCMC, see, e.g., Neal, 1993) methods usually based on Gibbs sampling (e.g., Hinton, 2002; Tieleman and Hinton, 2009; Desjardins et al., 2010b). The higher the mixing rate of the Markov chain, the fewer sampling steps are usually required for a proper MCMC approximation. For RBM learning algorithms it has been shown that the bias of the approximation increases with increasing absolute values of the model parameters (Bengio and Delalleau, 2009; Fischer and Igel, 2011a) and that this can indeed lead to severe distortions of the learning process (Fischer and Igel, 2010a). Thus, increasing the mixing rate of the Markov chains in RBM training is highly desirable.

In this paper, we propose to employ a Metropolis-type transition operator for RBMs that maximizes the probability of state changes in the framework of periodic sampling and can lead to a faster mixing Markov chain. This operator is related to the *Metropolized Gibbs* sampler introduced by Liu (1996) and the *flip-the-spin* operator with Metropolis acceptance rule used in Ising models (see related methods in section 7.3) and is, thus, referred to as *flip-the-state* operator. In contrast to these methods, our main theoretical result is that the proposed operator is also guaranteed to lead to an ergodic and thus properly converging Markov chain when using a periodic updating scheme (i.e., a deterministic scanning policy). It can replace Gibbs sampling in existing RBM learning algorithms without introducing computational overhead.

After a brief overview over RBM training and Gibbs sampling in section 7.2, section 7.3 introduces the flip-the-state transition operator and shows that the induced Markov chain converges to the RBM distribution. In section 7.4 we empirically analyze the mixing behavior of the proposed operator compared to Gibbs sampling by looking at the Second Largest Eigenvector Modulus (SLEM), the autocorrelation time, and the frequency of class changes in sample sequences. While the SLEM describes the speed of convergence to the equilibrium distribution, the autocorrelation time concerns the variance of an estimate when averaging over several successive samples of the Markov chain. The class changes quantify mixing between modes in our test problems. Furthermore, the effects of the proposed sampling procedure on learning in RBMs is studied. We discuss the results and conclude in Sections 7.5 and 7.6.

7.2 Background

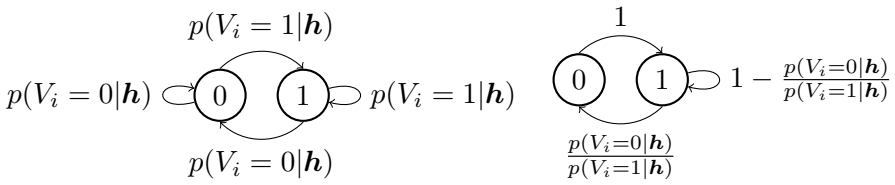
An RBM is an undirected graphical model with a bipartite structure (Smolensky, 1986; Hinton, 2002) consisting of one layer of m visible variables $\boldsymbol{V} = (V_1, \ldots, V_m)$ and one layer of n hidden variables $\boldsymbol{H} = (H_1, \ldots, H_n)$ taking values $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega := \{0,1\}^{m+n}$. The modeled joint distribution is $p(\boldsymbol{v}, \boldsymbol{h}) = e^{-E(\boldsymbol{v},\boldsymbol{h})} / \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ with energy E given by $E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$ with weights w_{ij} and biases b_j and c_i for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, jointly denoted as $\boldsymbol{\theta}$. By $\boldsymbol{v}_{-i}$ and $\boldsymbol{h}_{-i}$ we denote the vectors of the states of all visible and hidden variables, respectively, except the ith one.

Typical RBM training algorithms perform steepest ascent on approximations of the log-likelihood gradient. One of the most popular is Contrastive Divergence (CD, Hinton, 2002), which approximates the gradient of the log-likelihood by $-\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$, where $\boldsymbol{v}^{(k)}$ is a sample gained after k steps of Gibbs sampling starting from a training example $\boldsymbol{v}^{(0)}$.

Several variants of CD have been proposed. For example, in Persistent Contrastive Divergence (PCD, Tieleman, 2008) and its refinement Fast PCD (Tieleman and Hinton, 2009) the Gibbs chain is not initialized by a training example but maintains its current value between approximation steps. Parallel Tempering (PT, also known as replica exchange Monte Carlo sampling) has also been applied to RBMs (Cho et al., 2010; Desjardins et al., 2010b; Salakhutdinov, 2009). It introduces supplementary Gibbs chains that sample from more and more smoothed variants of the true probability distribution and allows samples to swap between chains. This leads to faster mixing, but introduces computational overhead.

In general, a homogeneous Markov chain on a finite state space Ω with N elements can be described by an $N \times N$ transition probability matrix $\boldsymbol{A} = (a_{\boldsymbol{x},\boldsymbol{y}})_{\boldsymbol{x},\boldsymbol{y}\in\Omega}$, where $a_{\boldsymbol{x},\boldsymbol{y}}$ is the probability that the Markov chain being in state $\boldsymbol{x}$ changes its state to $\boldsymbol{y}$ in the next time step. We denote the one step transition probability $a_{\boldsymbol{x},\boldsymbol{y}}$ by $A(\boldsymbol{x}, \boldsymbol{y})$, the n-step transition probability (the corresponding entry of the matrix $\boldsymbol{A}^n$) by $A^n(\boldsymbol{x}, \boldsymbol{y})$. The transition matrices are also referred to as transition operators. We write $\boldsymbol{p}$ for the N-dimensional probability vector corresponding to some distribution p over Ω.

When performing periodic Gibbs sampling in RBMs, we visit all hidden and all visible variables alternately in a block-wise fashion and update them according to their conditional probability given the state of the other layer (i.e., $p(h_i|\boldsymbol{v})$, $i = 1, ..., n$ and $p(v_j|\boldsymbol{h})$, $j = 1, ..., m$, respectively). Thus, the Gibbs transition operator $\boldsymbol{G}$ can be decomposed into two operators $\boldsymbol{G}_h$ and $\boldsymbol{G}_v$ (with $\boldsymbol{G} = \boldsymbol{G}_h \boldsymbol{G}_v$) changing only the state of the hidden layer or the visible layer, respectively. The two operators can be further decomposed into a set of basic transition operators $\boldsymbol{G}_k, k = 1, \ldots, (n+m)$, each



(a) Gibbs sampling (b) flip-the-state operator

Figure 7.1: Transition diagrams for a single variable V_i (a) updated by Gibbs sampling (b) updated by the flip-the-state transition operator (here $p(V_i = 0|\boldsymbol{h}) < p(V_i = 1|\boldsymbol{h})$).

updating just a single variable based on the conditional probabilities. An example of such a transition of a single variable based on these probabilities is depicted in the transition diagram in Figure 7.1(a).

7.3 The flip-the-state transition operator

In order to increase the mixing rate of the Markov chain, it seems desirable to change the basic transition operator of the Gibbs sampler in such a way that each single variable tends to change its state rather than sticking to the same state. This can be done by making the sample probability of a single neuron dependent on its current state. Transferring this idea to the transition graph shown in Figure 7.1, this means that we wish to decrease the probabilities associated to the self-loops and increase the transition probabilities between different states as much as possible. Of course, we have to ensure that the resulting Markov chain remains ergodic and still has the RBM distribution p as equilibrium distribution.

The transition probabilities are maximized by scaling the probability for a single variable to change from the less probable state to the more probable state to one (making this transition deterministic) while increasing the transition in the reverse direction accordingly with the same factor. In the – in practice not relevant but for theoretical considerations important – case of two states with the exact same conditional probability, we use the transition probabilities of Gibbs sampling to avoid a non-ergodic Markov chain.

These considerations can be formalized by first defining a variable v_i^* that indicates what the most probable state of the random variable V_i is or if both states are equally

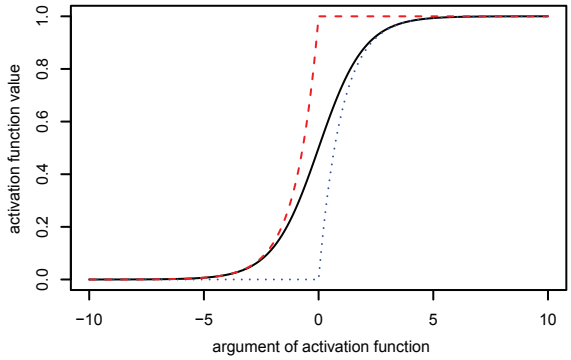


Figure 7.2: Activation function for Gibbs sampling (black) and for transitions based on $\boldsymbol{T}$, when the current state is 0 (red, dashed) or 1 (blue, dotted).

probable:

$$v_i^* = \begin{cases} 1 & \text{, if } p(V_i = 1|\boldsymbol{h}) > p(V_i = 0|\boldsymbol{h}) \\ 0 & \text{, if } p(V_i = 1|\boldsymbol{h}) < p(V_i = 0|\boldsymbol{h}) \\ -1 & \text{, if } p(V_i = 1|\boldsymbol{h}) = p(V_i = 0|\boldsymbol{h}) \end{cases} \tag{7.1}$$

Now we define the flip-the-state transition operator $\boldsymbol{T}$ as follows:

Definition 7.1. *For $i = 1, \ldots, m$, let the basic transition operator $\boldsymbol{T}_i$ for the visible unit V_i be defined through its transition probabilities: $T_i((\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}', \boldsymbol{h}')) = 0$ if $(\boldsymbol{v}, \boldsymbol{h})$ and $(\boldsymbol{v}', \boldsymbol{h}')$ differ in another variable than V_i and as*

$$T_i\big((\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}', \boldsymbol{h}')\big) = \begin{cases} \frac{p(v_i'|h)}{p(v_i|h)} & \text{, if } v_i^* = v_i \neq v_i' \\ 1 - \frac{p(v_i'|h)}{p(v_i|h)} & \text{, if } v_i^* = v_i = v_i' \\ 1 & \text{, if } v_i \neq v_i' = v_i^* \\ 0 & \text{, if } v_i = v_i' \neq v_i^* \\ \frac{1}{2} & \text{, if } v_i^* = -1 \end{cases} \tag{7.2}$$

otherwise. The transition matrix containing the transition probabilities of the visible layer is given by $\boldsymbol{T}_v = \prod_i \boldsymbol{T}_i$. The transition matrix for the hidden layer $\boldsymbol{T}_h$ is defined analogously, and the flip-the-state transition operator is given by $\boldsymbol{T} = \boldsymbol{T}_h \boldsymbol{T}_v$.

7.3.1 Activation function & computational complexity

An RBM corresponds to a stochastic, recurrent neural network with activation function $\sigma(x) = 1/(1 + e^{-x})$. Similarly, the transition probabilities defined in (7.2) can be

interpreted as resulting from an activation function depending not only on the weighted input to a neuron and its bias but also on the neuron's current state V_j (or analogously H_i):

$$\sigma'(x) = \begin{cases} \min\{e^x, 1\} & \text{if } V_j = 0 \\ \max\{1 - e^{-x}, 0\} & \text{if } V_j = 1 \end{cases} \tag{7.3}$$

Corresponding graphs are shown in Figure 7.2.

The differences in computational complexity between the activation functions σ and σ' can be neglected. The transition operator described here requires a switch based on the current state of the neuron on the one hand, but saves the computationally expensive call to the random generator in deterministic transitions on the other hand. Furthermore, in the asymptotic case, the running time of a sampling step is dominated by the matrix multiplications, while the number of activation function evaluations in one step increases only linearly with the number of neurons.

If the absolute value of the sum of the weighted inputs and the bias is large (i.e., extreme high conditional probability for one of the two states), the transition probabilities between states under Gibbs sampling are already almost deterministic. Thus, the difference between $\boldsymbol{G}$ and $\boldsymbol{T}$ decreases in this case. This is illustrated in Figure 7.2.

7.3.2 Related work

Both $\boldsymbol{G}$ and $\boldsymbol{T}$ are (local) Metropolis algorithms (Neal, 1993). A Metropolis algorithm proposes states with a proposal distribution and accepts them in a way which ensures detailed balance. In this view, Gibbs sampling corresponds to using the proposal distribution "flip current state" and the Boltzmann acceptance probability $\frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})+p(\boldsymbol{x}')}$, where $\boldsymbol{x}$ and $\boldsymbol{x}'$ denote the current and the proposed state, respectively. This proposal distribution has also been used with the Metropolis acceptance probability $\min\left(1, \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})}\right)$ for sampling from Ising models. The differences between the two acceptance functions are discussed, for example, by Neal (1993). He comes to the conclusion that "the issues still remain unclear, though it appears that common opinion favours using the Metropolis acceptance function in most circumstances" (p. 69).

The work by Peskun (1973) and Liu (1996) shows that the Metropolis acceptance function is optimal with respect to the asymptotic variance of the Monte Carlo estimate of the quantity of interest. This result only holds if the variables to be updated are picked randomly in each step of the (local) algorithm. Thus, they are not applicable in the typical RBM training scenario, where block-wise sampling in a predefined order is used. In this scenario, it can indeed happen that the flip-the-state proposal combined with the Metropolis acceptance function leads to non-ergodic chains as shown by the counter-examples given by Neal (1993, p. 69).

The transition operator $\boldsymbol{T}$ also uses the Metropolis acceptance probability, but the proposal distribution differs from the one used in Ising models in one detail, namely that it selects a state at random if the conditional probabilities of both states are equal. This is important from a theoretical point of view, because it ensures ergodicity as proven in the next section. This is the reason why our method does not suffer from the problems mentioned above.

Furthermore, Breuleux et al. (2011) discuss a similar idea to the one underlying our transition operator as a theoretic framework for understanding fast mixing, where one increases the probability to change states by defining a new transition matrix $\boldsymbol{A}'$ based on an existing transition matrix $\boldsymbol{A}$ by $\boldsymbol{A}' = (\boldsymbol{A} - \lambda \boldsymbol{I})(1 - \lambda)^{-1}$, where $\lambda \leq \min_{\boldsymbol{x} \in \Omega} A(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{I}$ is the identity matrix. Our method corresponds to applying this kind of transformation, not to the whole transition matrix, but rather to the transition probabilities of a single binary variable (i.e., the base transition operator). This makes the method not only computationally feasible in practice, but even more effective, because it allows us to redistribute more probability mass (because the redistribution is not limited by $\min_{\boldsymbol{x} \in \Omega} A(\boldsymbol{x}, \boldsymbol{x})$), so that more than one entry of the new transition matrix is 0.

7.3.3 Properties of the transition operator

To prove that a Markov chain based on the suggested transition operator $\boldsymbol{T}$ converges to the probability distribution p defined by the RBM, it has to be shown that p is invariant with respect to $\boldsymbol{T}$ and that the Markov chain is irreducible and aperiodic.

As stated above, the described transition operator belongs to the class of local Metropolis algorithms. This implies that detailed balance holds for all the base transition operators (see, e.g., Neal, 1993). If p is invariant w.r.t the basic transition operators it is also invariant w.r.t. the concatenated transition matrix $\boldsymbol{T}$.

However, there is no general proof of ergodicity of Metropolis algorithms if neither the proposal distribution nor the acceptance distribution are strictly positive and the base transitions are applied deterministically in a fixed order. Therefore irreducibility and aperiodicity still remain to be proven (see, e.g., Neal, 1993, p. 56).

To show irreducibility, we need some definitions and a lemma first. For a fixed hidden state $\boldsymbol{h}$ let us define $\boldsymbol{v}_{\max}(\boldsymbol{h})$ as the visible state that maximizes the probability of the whole state,

$$\boldsymbol{v}_{\max}(\boldsymbol{h}) := \arg\max_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{h}) \ , \tag{7.4}$$

and analogously

$$\boldsymbol{h}_{\max}(\boldsymbol{v}) := \arg\max_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \ . \tag{7.5}$$

We assume that arg max is unique and that ties are broken by taking the greater state according to some arbitrary predefined strict total order $\prec$.

Furthermore, let $\mathcal{M}$ be the set of states, for which the probability can not be increased by changing either only the hidden or only the visible states:

$$\mathcal{M} = \left\{ \big(\boldsymbol{v}, \boldsymbol{h}\big) \in \Omega \,\middle|\, \big(\boldsymbol{v}, \boldsymbol{h}\big) = \big(\boldsymbol{v}_{\max}(\boldsymbol{h}), \boldsymbol{h}\big) = \big(\boldsymbol{v}, \boldsymbol{h}_{\max}(\boldsymbol{v})\big) \right\} \tag{7.6}$$

Note, that $\mathcal{M}$ is not the empty set, since it contains at least the most probable state $\arg\max_{(\boldsymbol{v},\boldsymbol{h})} p(\boldsymbol{v}, \boldsymbol{h})$. Now we have:

Lemma 7.1. *From every state $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega$ one can reach $\big(\boldsymbol{v}_{\max}(\boldsymbol{h}), \boldsymbol{h}\big)$ by applying the visible transition operator $\boldsymbol{T}_v$ once and $\big(\boldsymbol{v}, \boldsymbol{h}_{\max}(\boldsymbol{v})\big)$ in one step of $\boldsymbol{T}_h$. It is possible to reach every state $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega$ in one step of $\boldsymbol{T}_v$ from $\big(\boldsymbol{v}_{\max}(\boldsymbol{h}), \boldsymbol{h}\big)$ and in one step of $\boldsymbol{T}_h$ from $\big(\boldsymbol{v}, \boldsymbol{h}_{\max}(\boldsymbol{v})\big)$.*

Proof. From the definition of $\boldsymbol{v}_{\max}(\boldsymbol{h})$ and the independence of the conditional probabilities of the visible variables given the state of the hidden layer it follows:

$$p\big(\boldsymbol{v}_{\max}(\boldsymbol{h})|\boldsymbol{h}\big) = \max_{v_1,\dots,v_n} \prod_i p(v_i|\boldsymbol{h}) \ . \tag{7.7}$$

Thus, in $\boldsymbol{v}_{\max}(\boldsymbol{h})$ every single visible variable is in the state with the higher conditional probability (i.e., in v_i^*) or both states are equally probable (in which case $v_i^* = -1$). By looking at the definition of the base transitions (7.2) it becomes clear that this means that $T_i\big((\boldsymbol{v}, \boldsymbol{h}), (v_{\max}(\boldsymbol{h})_i, \boldsymbol{v}_{-i}, \boldsymbol{h})\big) > 0$ and $T_i\big((\boldsymbol{v}_{\max}(\boldsymbol{h}), \boldsymbol{h}), (v_i, \boldsymbol{v}_{\max}(\boldsymbol{h})_{-i}, \boldsymbol{h})\big) > 0$. So we get for all $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega$:

$$T_v\big((\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}_{\max}(\boldsymbol{h}), \boldsymbol{h})\big) = \prod_i T_i\big((\boldsymbol{v}, \boldsymbol{h}), (v_{\max}(\boldsymbol{h})_i, \boldsymbol{v}_{-i}, \boldsymbol{h})\big) > 0 \tag{7.8}$$

$$T_v\big((\boldsymbol{v}_{\max}(\boldsymbol{h}), \boldsymbol{h}), (\boldsymbol{v}, \boldsymbol{h})\big) = \prod_i T_i\big((\boldsymbol{v}_{\max}(\boldsymbol{h}), \boldsymbol{h}), (v_i, \boldsymbol{v}_{\max}(\boldsymbol{h})_{-i}, \boldsymbol{h})\big) > 0 \tag{7.9}$$

This holds equivalently for the hidden transition operator $\boldsymbol{T}_h$ and $(\boldsymbol{v}, \boldsymbol{h}_{\max}(\boldsymbol{v}))$. For all $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega$:

$$T_h\big((\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}, \boldsymbol{h}_{\max}(\boldsymbol{v}))\big) = \prod_i T_i\big((\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}, h_{\max}(\boldsymbol{v})_i, \boldsymbol{h}_{-i})\big) > 0 \tag{7.10}$$

$$T_h\big((\boldsymbol{v}, \boldsymbol{h}_{\max}(\boldsymbol{v})), (\boldsymbol{v}, \boldsymbol{h})\big) = \prod_i T_i\big((\boldsymbol{v}, \boldsymbol{h}_{\max}(\boldsymbol{v})), (\boldsymbol{v}, h_i, \boldsymbol{h}_{\max}(\boldsymbol{v})_{-i})\big) > 0 \tag{7.11}$$

□

Now we prove the irreducibility:

Theorem 7.1. *The Markov chain induced by $\boldsymbol{T}$ is irreducible:*

$$\forall (\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}', \boldsymbol{h}') \in \Omega : \exists n > 0 : T^n\big((\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}', \boldsymbol{h}')\big) > 0 \tag{7.12}$$

Proof. The proof is divided into three steps showing:

(i) from every state $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega$ one can reach an element of $\mathcal{M}$ in a finite number of transitions, i.e., $\forall(\boldsymbol{v}, \boldsymbol{h}) \in \Omega\ \exists(\boldsymbol{v}^*, \boldsymbol{h}^*) \in \mathcal{M}$ and $n \in \mathbb{N}$, with $T^n\big((\boldsymbol{v}, \boldsymbol{h}), (\boldsymbol{v}^*, \boldsymbol{h}^*)\big) > 0$,

(ii) for every state $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega$ there exists a state $(\boldsymbol{v}^*, \boldsymbol{h}^*) \in \mathcal{M}$ from which it is possible to reach $(\boldsymbol{v}, \boldsymbol{h}) \in \Omega$ in a finite number of transitions, i.e., $\forall(\boldsymbol{v}, \boldsymbol{h}) \in \Omega\ \exists(\boldsymbol{v}^*, \boldsymbol{h}^*) \in \mathcal{M}$ and $n \in \mathbb{N}$ with $T^n\big((\boldsymbol{v}^*, \boldsymbol{h}^*), (\boldsymbol{v}, \boldsymbol{h})\big) > 0$, and

(iii) any transition between two arbitrary elements in $\mathcal{M}$ is possible, i.e., $\forall(\boldsymbol{v}^*, \boldsymbol{h}^*)$, $(\boldsymbol{v}^{**}, \boldsymbol{h}^{**}) \in \mathcal{M} : T\big((\boldsymbol{v}^*, \boldsymbol{h}^*), (\boldsymbol{v}^{**}, \boldsymbol{h}^{**})\big) > 0$.

Step (i): Let us define a sequence $\big((\boldsymbol{v}_k, \boldsymbol{h}_k)\big)_{k\in\mathbb{N}}$ with $\boldsymbol{v}_0 := \boldsymbol{v}$, $\boldsymbol{h}_0 := \boldsymbol{h}$ and $\boldsymbol{h}_k := \boldsymbol{h}_{\max}(\boldsymbol{v}_{k-1})$ and $\boldsymbol{v}_k := \boldsymbol{v}_{\max}(\boldsymbol{h}_k)$ for $k > 0$. From the definition of $\boldsymbol{v}_{\max}$ and $\boldsymbol{h}_{\max}$ it follows that $(\boldsymbol{v}_{k-1}, \boldsymbol{h}_{k-1}) \neq (\boldsymbol{v}_k, \boldsymbol{h}_k)$ unless $(\boldsymbol{v}_{k-1}, \boldsymbol{h}_{k-1}) \in \mathcal{M}$ and that no state in $\Omega \setminus \mathcal{M}$ is visited twice. The latter follows from the fact that in the sequence two successive states $(\boldsymbol{v}_k, \boldsymbol{h}_k)$ and $(\boldsymbol{v}_{k+1}, \boldsymbol{h}_{k+1})$ from $\Omega \setminus \mathcal{M}$ have either increasing probabilities or $(\boldsymbol{v}_k, \boldsymbol{h}_k) \prec (\boldsymbol{v}_{k+1}, \boldsymbol{h}_{k+1})$. Since Ω is a finite set, such a sequence must reach a state $(\boldsymbol{v}_n, \boldsymbol{h}_n) = (\boldsymbol{v}_{n+i}, \boldsymbol{h}_{n+i}) \in \mathcal{M}, i \in \mathbb{N}$ after a finite number of steps n.

Finally, this sequence can be produced by $\boldsymbol{T}$ since from eq. (7.8) and eq. (7.10) it follows that $\forall k > 0$:

$$T\big((\boldsymbol{v}_{k-1}, \boldsymbol{h}_{k-1}), (\boldsymbol{v}_k, \boldsymbol{h}_k)\big) = T_h\big((\boldsymbol{v}_{k-1}, \boldsymbol{h}_{k-1}), (\boldsymbol{v}_{k-1}, \boldsymbol{h}_k)\big) \cdot T_v\big((\boldsymbol{v}_{k-1}, \boldsymbol{h}_k), (\boldsymbol{v}_k, \boldsymbol{h}_k)\big) > 0 \quad (7.13)$$

Hence, one can get from $(\boldsymbol{v}_{k-1}, \boldsymbol{h}_{k-1})$ to $(\boldsymbol{v}_k, \boldsymbol{h}_k)$ in one step of the transition operator $\boldsymbol{T}$.

Step (ii) We now consider a similar sequence $\big((\boldsymbol{v}_k, \boldsymbol{h}_k)\big)_{k\in\mathbb{N}}$ with $\boldsymbol{v}_0 := \boldsymbol{v}$, $\boldsymbol{h}_0 := \boldsymbol{h}$ and $\boldsymbol{v}_k := \boldsymbol{v}_{\max}(\boldsymbol{h}_{k-1})$ and $\boldsymbol{h}_k := \boldsymbol{h}_{\max}(\boldsymbol{v}_k)$, for $k > 0$. Again, there exists $n \in \mathbb{N}$, so that $(\boldsymbol{v}_n, \boldsymbol{h}_n) = (\boldsymbol{v}_{n+i}, \boldsymbol{h}_{n+i}) \in \mathcal{M}, i \in \mathbb{N}$. From equations (7.9) and (7.11) it follows that $\forall k > 0$:

$$T\big((\boldsymbol{v}_k, \boldsymbol{h}_k), (\boldsymbol{v}_{k-1}, \boldsymbol{h}_{k-1})\big) = T_h\big((\boldsymbol{v}_k, \boldsymbol{h}_k), (\boldsymbol{v}_k, \boldsymbol{h}_{k-1})\big) \cdot T_v\big((\boldsymbol{v}_k, \boldsymbol{h}_{k-1}), (\boldsymbol{v}_{k-1}, \boldsymbol{h}_{k-1})\big) > 0 \quad (7.14)$$

That is, one can get from $(\boldsymbol{v}_{k+1}, \boldsymbol{h}_{k+1})$ to $(\boldsymbol{v}_k, \boldsymbol{h}_k)$ in one step of the transition operator $\boldsymbol{T}$ and follow the sequence backwards from $(\boldsymbol{v}_n, \boldsymbol{h}_n) \in \mathcal{M}$ to $(\boldsymbol{v}, \boldsymbol{h})$.

Step (iii) From equations (7.8)–(7.11) it follows directly that a transition between two arbitrary points in $\mathcal{M}$ is always possible. □

Showing the aperiodicity is straight-forward:

Theorem 7.2. *The Markov chain induced by $\boldsymbol{T}$ is aperiodic.*

Proof. For every state $(\boldsymbol{v}^*, \boldsymbol{h}^*)$ in the nonempty set $\mathcal{M}$ it holds that

$$T\big((\boldsymbol{v}^*, \boldsymbol{h}^*), (\boldsymbol{v}^*, \boldsymbol{h}^*)\big) > 0 \ , \tag{7.15}$$

so the state is aperiodic. This means that the whole Markov chain is aperiodic, since it is irreducible (see, e.g., Brémaud, 1999). □

Theorems 7.1 and 7.2 show that the Markov chain induced by the operator $\boldsymbol{T}$ has p as its equilibrium distribution, i.e., the Markov chain is ergodic with stationary distribution p.

7.4 Experiments

First, we experimentally compare the mixing behavior of the flip-the-state method with Gibbs sampling by analyzing $\boldsymbol{T}$ and $\boldsymbol{G}$ for random RBMs. Then, we study the effects of replacing $\boldsymbol{G}$ by $\boldsymbol{T}$ in different RBM learning algorithms applied to benchmark problems. After that, the operators are used to sample sequences from trained RBMs. The autocorrelation times and the number of class changes reflecting mode changes are compared. Training and sampling the RBMs was implemented using the open-source machine learning library Shark (Igel et al., 2008).

7.4.1 Analysis of the convergence rate

The convergence speed of an ergodic, homogeneous Markov chain with finite state space is governed by the second largest eigenvector modulus (SLEM). This is a direct consequence of the Perron-Frobenius theorem. Note that the SLEM computation considers absolute values, in contrast to the statements by Liu (1996) referring to the signed eigenvalues. We calculated the SLEM for transition matrices of Gibbs sampling and the new transition operator for small, randomly generated RBMs by solving the eigenvector equation of the resulting transition matrices $\boldsymbol{G}$ and $\boldsymbol{T}$. To handle the computational complexity we had to restrict our considerations to RBMs with only 2, 3, and 4 visible and hidden neurons, respectively. The weights of these RBMs were drawn randomly and uniformly from $[-c; c]$, with $c \in \{1, \ldots, 10\}$, and bias parameters were set to zero. For each value of c we generated 100 RBMs and compared the SLEMs of $\boldsymbol{G}$ and $\boldsymbol{T}$.

7.4.2 Log-likelihood evolution during training

We study the evolution of the exact log-likelihood, which is tractable if either the number of the hidden or the visible units is chosen to be small enough, during gradient-based training of RBMs using CD, PCD, or PT based on samples produced by Gibbs sampling and the flip-the-state transition operator.

We used three benchmark problems taken from the literature. Desjardins et al. (2010b) consider a parametrized artificial problem, referred to as *Artificial Modes* in the following, for studying mixing properties. The inputs are 4×4 binary images. The observations are distributed around four equally likely basic modes, from which samples are generated by flipping pixels. The probability of flipping a pixel is given by the parameter p_{mut}, controlling the "effective distance between each mode" (Desjardins et al., 2010b). In our experiments, p_{mut} was either 0.01 or 0.1. Furthermore, we used a 4×4 pixel version of *Bars and Stripes* (MacKay, 2002) and finally the MNIST data set of handwritten digits.

In the small toy problems (*Artificial Modes* and *Bars and Stripes*) the number of hidden units was set to be the same as the number of visible units, i.e., $n = 16$. For MNIST the number of hidden units was set to 10. The RBMs were initialized with weights and biases drawn uniformly from a Gaussian distribution with 0 mean and standard deviation 0.01.

The models were trained on all benchmark problems using gradient ascent on the gradient approximation of either CD or PCD with k sampling steps (which we refer to as CD_k or PCD_k) or PT. Note that CD learning with $k = 1$ does not seem to be a reasonable scenario for applying the new operator. The performance of PT depends on the number t of tempered chains and on the number of sampling steps k carried out in each tempered chain before swapping samples between chains. We call PT with t temperatures and k sampling steps t-PT_k. The inverse temperatures were distributed uniformly between 0 and 1. Samples for each learning method where either obtained by $\boldsymbol{G}$ or $\boldsymbol{T}$.

We performed mini-batch learning with a batch size of 100 training examples in the case of MNIST and *Artificial Modes* and batch learning for *Bars and Stripes*. The number of samples used for the gradient approximation was set to be equal to the number of training examples in a (mini) batch. We tested different learning rates $\eta \in \{0.01, 0.05, 0.1\}$ and used neither weight decay nor a momentum parameter. All experiments were run for a length of 20000 update steps and repeated 25 times. We calculated the log-likelihood every 100th step of training. In the following, all reported log-likelihood values are averaged over the training examples.

7.4.3 Autocorrelation analysis

To measure the mixing properties of the operators on larger RBMs, we performed an autocorrelation analysis.

We estimated the autocorrelation function

$$R(\Delta t) = \frac{\mathrm{cov}(\mathcal{E}(\boldsymbol{V}_k, \boldsymbol{H}_k)\mathcal{E}(\boldsymbol{V}_{k+\Delta t}, \boldsymbol{H}_{k+\Delta t}))}{\mathrm{var}(\mathcal{E}(\boldsymbol{V}_k, \boldsymbol{H}_k))} \ . \tag{7.16}$$

The random variables $\boldsymbol{V}_k$ and $\boldsymbol{H}_k$ are the state of the visible and hidden variables after running the chain for k steps. The chain is assumed to be stationary, which induces $\mathbb{E}[\mathcal{E}(\boldsymbol{V}_k, \boldsymbol{H}_k)] = \mathbb{E}[\mathcal{E}(\boldsymbol{V}_{k+\Delta t}, \boldsymbol{H}_{k+\Delta t})]$ and $\mathrm{var}(\mathcal{E}(\boldsymbol{V}_k, \boldsymbol{H}_k)) = \mathrm{var}(\mathcal{E}(\boldsymbol{V}_{k+\Delta t}, \boldsymbol{H}_{k+\Delta t}))$. The autocorrelation function is always defined with respect to a specific function on the state space. Here the energy function $\mathcal{E}$ is a natural choice.

The autocorrelation time is linked to the asymptotic variance of an estimator based on averaging over consecutive samples from a Markov chain. It is defined as

$$\tau = \sum_{\Delta t=-\infty}^{\infty} R(\Delta t) \ . \tag{7.17}$$

An estimator based on $l\tau$ consecutive samples from a Markov chain has the same variance as an estimator based on l independent samples (see, e.g., Neal, 1993). In this sense τ consecutive samples are equivalent to one independent sample.

For the autocorrelation experiments we trained 25 RBMs on each of the previously mentioned benchmark problems with 20-PT$_{10}$. In addition to the RBMs with 10 hidden units we trained 24 RBMs with 500 hidden neurons on MNIST for 2000 parameter updates. To estimate the autocorrelations we sampled these RBMs for one million steps using $\boldsymbol{G}$ and $\boldsymbol{T}$, respectively. We followed the recommendations by Thompson (2010) and, in addition to calculating and plotting the autocorrelations directly, fitted AR-models to the times series to estimate the autocorrelation time using the software package SamplerCompare (Thompson, 2011).

7.4.4 Frequency of class changes

To access the ability of the two operators to mix between different modes, we observed the class changes in sample sequences, similar to the experiments by Bengio et al. (2013). We trained 25 RBMs with CD-5 on *Artificial Modes* with $p_{\mathrm{mut}} = 0.01$ and $p_{\mathrm{mut}} = 0.1$. After training, we sampled from the RBMs using either $\boldsymbol{T}$ or $\boldsymbol{G}$ as transition operator and analyzed how often subsequent samples belong to different classes. We considered four classes. Each class was defined by one of the four basic modes used to generate the dataset. A sample belongs to the same class as the mode to which it has the smallest Hamming distance. Ambiguous samples which could not be assigned

to a single class, because they were equally close to at least two of the modes, were discarded. In one experimental setting, all trained RBMs were initialized 1000 times with samples drawn randomly from the training distribution (representing the starting distribution of CD learning), and the number of sampling steps before the first class change was measured. In a second setting, for each RBM one chain was started with all visible units set to one and run for 10000 steps. Afterwards, the number of class changes was counted.

7.5 Results and discussion

7.5.1 Analysis of the convergence rate

The upper plot in Figure 7.3 shows the fraction of RBMs (out of 100) for which the corresponding transition operator $\boldsymbol{T}$ has a smaller SLEM than the Gibbs operator $\boldsymbol{G}$ (and therefore $\boldsymbol{T}$ promises a faster mixing Markov chain than $\boldsymbol{G}$) in dependence on the value of c, which upper bounds the weights. If all the weights are equal to zero, Gibbs sampling is always better, but the higher the weights get the more often $\boldsymbol{T}$ has a better mixing rate. This effect is the more pronounced the more neurons the RBM has, which suggests that the results of our analysis can be transfered to real world RBMs.

In the hypothetical case that all variables are independent (corresponding to an RBM where all weights are zero), Gibbs sampling is optimal and converges in a single step. With the flip-the-state operator, however, the probability of a neuron to be in a certain state would oscillate and converge exponentially by a factor of $\frac{1-p(v_i^*)}{p(v_i^*)}$ (i.e., the SLEM of the base transition matrix in this case) to the equilibrium distribution. As the variables get more and more dependent, the behavior of Gibbs sampling is no longer optimal and the Gibbs chain converges more slowly than the Markov chain induced by $\boldsymbol{T}$. Figure 7.3 directly supports our claim that in this relevant scenario changing states more frequently by the flip-the-state method can improve mixing.

7.5.2 Log-likelihood evolution during training

To summarize all trials of one experiment into a single value we calculated the maximum log-likelihood value reached during each run and finally calculated the median over all runs. The resulting maximum log-likelihood values for different experimental settings for learning the *Bars and Stripes* and the MNIST data set with CD and PT are shown in Table 7.1. Similar results were found for PCD and for experiments on *Artificial Modes*, see appendix. For most experimental settings, the RBMs reaches statistically significant higher likelihood values during training with the new transition operator (Wilcoxon signed-rank test, $p < 0.05$).

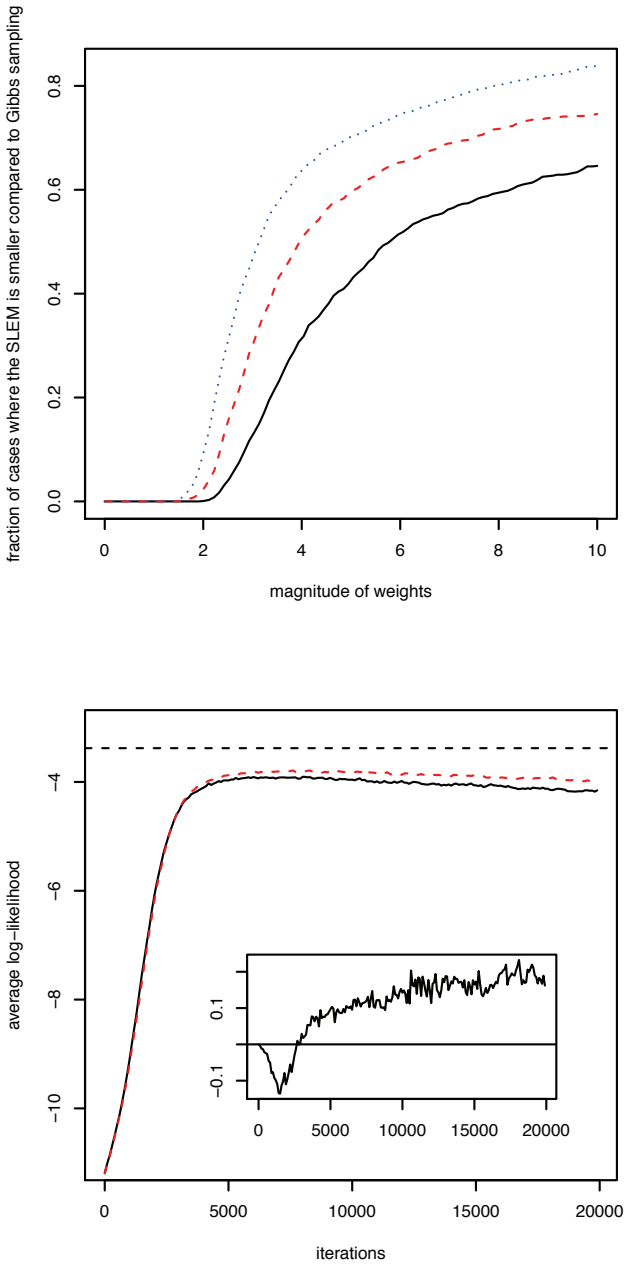


Figure 7.3: The upper figure compares the mixing rates of $\boldsymbol{G}$ and $\boldsymbol{T}$ for 2×2 RBMs (black), 3×3 RBMs (red, dashed) and 4×4 RBMs (blue, dotted). The lower figure depicts the learning curves for CD_5 on *Bars and Stripes* with learning rate 0.05 using $\boldsymbol{G}$ (black) or $\boldsymbol{T}$ (red, dashed). The inset shows the difference between the two and is positive if the red curve is higher. The dashed horizontal line indicates the maximum possible value of the average log-likelihood.

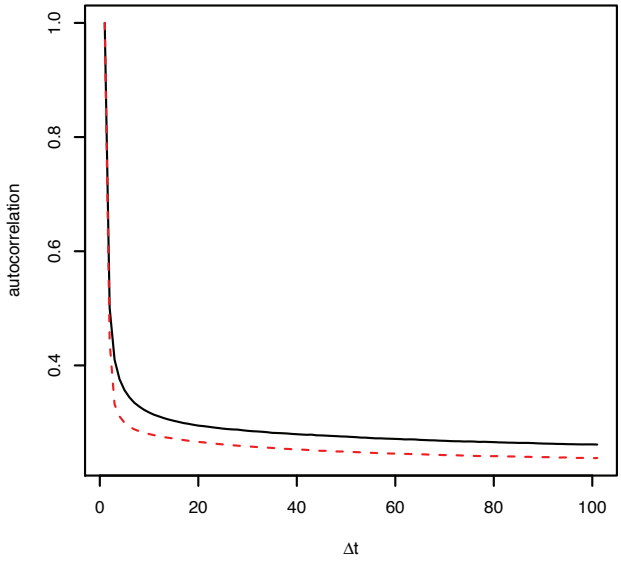


Figure 7.4: Autocorrelation function $R(\Delta t)$ for RBMs with 500 hidden neurons trained on MNIST based on 24 trials, sampled 10^6 steps each. The dotted line corresponds to $\boldsymbol{T}$ and the solid one to $\boldsymbol{G}$.

If we examine the evolution of likelihood values over time (as shown, e.g., in the lower plot of Figure 7.3) more closely, we see that the proposed transition operator is better in the end of training, but Gibbs sampling is actually slightly better in the beginning when weights are close to their small initialization. Learning curves as in Figure 7.3 also show that if divergence occurs with Gibbs sampling (Fischer and Igel, 2010a), it will be slowed down, but not completely avoided with the new transition operator.

It is not surprising that Gibbs sampling mixes better at the beginning of the training, because the variables are almost independent when the weights are still close to their initial values near zero. Still, the results confirm that the proposed transition operator mixes better in the difficult phase of RBM training and that the faster mixing helps reaching better learning results.

The results suggest that it may be reasonable to mix the two operators. Either, one could start with $\boldsymbol{G}$ and switch to $\boldsymbol{T}$ as the weights grow larger, or one can softly blend between the basic operators and consider $\boldsymbol{T}_i^{\alpha} = \alpha \boldsymbol{T}_i + (1-\alpha)\boldsymbol{G}_i$, $\alpha \in [0,1]$.

Table 7.1: Median maximum log-likelihood values on *Bars and Stripes* (top) and MNIST (bottom). Significant differences are marked with a star.

Bars and Stripes			
Algorithm	η	Gibbs	$\boldsymbol{T}$
CD_5	0.01	-4.070406	-3.986813*
CD_5	0.05	-3.832875	-3.727781*
CD_5	0.1	-3.838406	-3.732438*
CD_{10}	0.01	-3.963563	-3.930687*
CD_{10}	0.05	-3.640219	-3.57625*
CD_{10}	0.1	-3.635781	-3.589219*
5-PT_1	0.01	-4.011406	-4.0095
5-PT_1	0.05	-3.675312	-3.636125*
5-PT_1	0.1	-3.8255	-3.77825
5-PT_5	0.01	-3.928125	-3.918781
5-PT_5	0.05	-3.515719	-3.500844*
5-PT_5	0.1	-3.565281	-3.540625*
20-PT_1	0.01	-3.974219	-3.977562
20-PT_1	0.05	-3.548969	-3.524406*
20-PT_1	0.1	-3.577812	-3.549812*
20-PT_5	0.01	-3.917969	-3.923781
20-PT_5	0.05	-3.470094	-3.466188*
20-PT_5	0.1	-3.478844	-3.472594*

MNIST			
Algorithm	η	Gibbs	$\boldsymbol{T}$
CD_5	0.01	-178.716	-177.958*
CD_5	0.05	-179.345	-178.873
CD_5	0.1	-179.007	-178.446*
CD_{10}	0.01	-176.495	-175.638*
CD_{10}	0.05	-176.844	-176.476
CD_{10}	0.1	-177.925	-176.586
10-PT_2	0.01	-182.283	-180.272*
10-PT_2	0.05	-182.303	-181.379*
10-PT_2	0.1	-181.727	-180.164
10-PT_5	0.01	-178.71	-178.215*
10-PT_5	0.05	-179.625	-178.708*
10-PT_5	0.1	-179.051	-178.504*

Table 7.2: Mean autocorrelation times $\tau_{\boldsymbol{G}}$ and $\tau_{\boldsymbol{T}}$ for single Markov chains using the Gibbs sampler and the flip-the-state operator. The last column shows the gain defined as $1 - \frac{\tau_{\boldsymbol{T}}}{\tau_{\boldsymbol{G}}}$.

	Gibbs $\tau_{\boldsymbol{G}}$	$\boldsymbol{T}$ $\tau_{\boldsymbol{T}}$	gain in %
Bars and Stripes	22.46	20.06	10.67
Artificial Modes, $p_{\text{mut}} = 0.1$	3.16	2.19	30.73
Artificial Modes, $p_{\text{mut}} = 0.01$	6.00	5.94	1.06
MNIST, $n = 10$	488.26	445.84	8.69
MNIST, $n = 500$	522.39	432.12	17.28

7.5.3 Autocorrelation analysis

The autocorrelation analysis revealed that sampling using the flip-the-state operator leads to shorter autocorrelation times in the considered benchmark problems, see Table 7.2 and Figure 7.4. For example, an RBM trained on MNIST with 500 hidden neurons needed on average to be sampled for 17.28% fewer steps to achieve the same variance of the estimate if $\boldsymbol{T}$ is used instead of $\boldsymbol{G}$ – without overhead in computation time or implementation complexity. The results with $n = 500$ demonstrate that our previous findings carry over to larger RBMs.

7.5.4 Frequency of class changes

The numbers of class changes observed in sequences of 10000 samples starting from the visible nodes set to one produced by $\boldsymbol{G}$ and $\boldsymbol{T}$ are given in Table 7.3.

Table 7.3: Frequencies of class changes for the Gibbs sampler and the flip-the-state operator in sequences of 10000 samples (medians and quantiles over samples from 25 RBMs).

	25% quantile	median	75% quantile
Artificial Modes, $p_{\text{mut}} = 0.1$, $\boldsymbol{G}$	615	637	655
Artificial Modes, $p_{\text{mut}} = 0.1$, $\boldsymbol{T}$	919	944	958
Artificial Modes, $p_{\text{mut}} = 0.01$, $\boldsymbol{G}$	134	148	162
Artificial Modes, $p_{\text{mut}} = 0.01$, $\boldsymbol{T}$	175	186	199

Table 7.4 shows the number of samples before the first class change when initializing the Markov chain with samples randomly drawn from the training distribution. Markov chains based on $\boldsymbol{T}$ led to more and faster class changes than chains using Gibbs sampling. As the modes in the training set get more distinct (comparing $p_{\text{mut}} = 0.1$

to $p_{\text{mut}} = 0.01$) class changes get less frequent and more sampling steps are needed to yield a class change. Nevertheless, $\boldsymbol{T}$ is superior to $\boldsymbol{G}$ even in this setting.

Table 7.4: Number of samples before the first class change when starting a Markov chain with samples from the training distribution (medians and quantiles over samples from 25 RBMs).

	25% quantile	median	75% quantile
Artificial Modes, $p_{\text{mut}} = 0.1$, $\boldsymbol{G}$	6	13	25
Artificial Modes, $p_{\text{mut}} = 0.1$, $\boldsymbol{T}$	3	7	14
Artificial Modes, $p_{\text{mut}} = 0.01$, $\boldsymbol{G}$	16	41	96
Artificial Modes, $p_{\text{mut}} = 0.01$, $\boldsymbol{T}$	10	27	63

7.6 Conclusion

We proposed the flip-the-state transition operator for MCMC-based training of RBMs and proved that it induces a converging Markov chain. Large weights lead to slow mixing Gibbs chains that can severely harm RBM training. In this scenario, the proposed flip-the-state method increases the mixing rate compared to Gibbs sampling. The way of sampling is generally applicable in the sense that it can be employed in every learning method for binary RBMs relying on Gibbs sampling, for example contrastive divergence learning and its variants as well as Parallel Tempering. As empirically shown, the better mixing indeed leads to better learning results in practice. As the flip-the-state sampling does not introduce computational overhead, we see no reason to stick to standard Gibbs sampling.

7.7 Appendix

Log-likelihood values for different problems, algorithms, and experimental settings

Table 7.5: Median maximum log-likelihood values for different experimental settings for learning *Bars and Stripes* (left) and MNIST (right). Significant differences are marked with a star.

Bars and Stripes			
Algorithm	η	Gibbs	$\boldsymbol{T}$
PCD_1	0.01	-4.944813*	-5.131875
PCD_1	0.05	-4.917219	-4.754625*
PCD_1	0.1	-5.285469	-5.176563*
PCD_5	0.01	-4.067437	-4.000844*
PCD_5	0.05	-4.050906	-3.915938*
PCD_5	0.1	-4.209375	-4.124625*
PCD_{10}	0.01	-3.972812	-3.945219*
PCD_{10}	0.05	-3.8425	-3.769938*
PCD_{10}	0.1	-4.02	-3.917937*

MNIST			
Algorithm	η	Gibbs	$\boldsymbol{T}$
PCD_1	0.01	-185.536	-185.378*
PCD_1	0.05	-181.382	-180.905
PCD_1	0.1	-179.572	-180.502
PCD_5	0.01	-179.061	-177.939*
PCD_5	0.05	-178.195	-177.675
PCD_5	0.1	-176.897	-175.946
PCD_{10}	0.01	-175.799	-174.919*
PCD_{10}	0.05	-176.301	-175.446*
PCD_{10}	0.1	-176.208	-174.94

Table 7.6: Median maximum log-likelihood values for different experimental settings for learning *Artificial Modes*. The left table shows the results for datasets generated with a probability p_{mut} of permuting each pixel of 0.1, the right table for $p_{\text{mut}} = 0.01$. Significant differences are marked with a star.

Artificial Modes, $p_{\text{mut}} = 0.1$			
Algorithm	η	Gibbs	$\boldsymbol{T}$
CD_5	0.01	-6.79103	-6.78603*
CD_5	0.05	-6.80241	-6.79473*
CD_{10}	0.01	-6.78564	-6.7833*
CD_{10}	0.05	-6.79646	-6.79682*
PCD_5	0.01	-6.79176	-6.78537*
PCD_5	0.05	-6.80292	-6.79679*
PCD_{10}	0.01	-6.78512	-6.78329*
PCD_{10}	0.05	-6.79575	-6.79372*
10-PT_2	0.01	-6.78282	-6.78325
10-PT_2	0.05	-6.79839	-6.79575
10-PT_5	0.01	-6.78206	-6.78213
10-PT_5	0.05	-6.7929	-6.79351
10-PT_{10}	0.01	-6.7827	-6.78203
10-PT_{10}	0.05	-6.79101	-6.79196

Artificial Modes, $p_{\text{mut}} = 0.01$			
Algorithm	η	Gibbs	$\boldsymbol{T}$
CD_5	0.01	-4.01644	-3.64562*
CD_5	0.05	-4.00452	-3.68796*
CD_{10}	0.01	-3.48928	-3.23056*
CD_{10}	0.05	-3.51262	-3.27728*
PCD_5	0.01	-3.9956	-3.6295*
PCD_5	0.05	-3.94864	-3.61392*
PCD_{10}	0.01	-3.46321	-3.21007*
PCD_{10}	0.05	-3.37534	-3.11163*
10-PT_2	0.01	-2.40041	-2.40195
10-PT_2	0.05	-2.40682	-2.40798
10-PT_5	0.01	-2.39929	-2.3992
10-PT_5	0.05	-2.40648	-2.40072*
10-PT_{10}	0.01	-2.39973	-2.40012
10-PT_{10}	0.05	-2.40188	-2.40078

Chapter 8

How to center binary RBMs

This chapter is based on the manuscript "How to center binary restricted Boltzmann machines" by J. Melchior, A. Fischer, and L. Wiskott, submitted.

Abstract

This work analyzes centered binary Restricted Boltzmann Machines (RBMs), where centering is done by subtracting offset values from visible and hidden variables. We show analytically that (i) centering can be reformulated as a different update rule for normal binary RBMs, (ii) the expected performance of centered binary RBMs is invariant under simultaneous flip of data and offsets, for any offset value in the range of zero to one, and (iii) using the enhanced gradient is equivalent to setting the offset values to the average over model and data mean. Due to the structural similarity this results also generalize to deep Boltzmann machines. Furthermore, numerical simulations suggest that (i) optimal generative performance is achieved by subtracting mean values from visible as well as hidden variables, (ii) centered RBMs reach significantly higher log-likelihood values than normal binary RBMs, (iii) the enhanced gradient suffers from divergence more often than other centering variants, (iv) learning is stabilized if an exponentially moving average over the batch means is used for the offset values instead of the current batch mean, which also prevents the enhanced gradient from diverging, and (v) centering leads to an update direction that is closer to the natural gradient.

8.1 Introduction

In the last decade Restricted Boltzmann Machines (RBMs) got into the focus of attention because they can be considered as building blocks of deep neural networks (Hinton et al., 2006; Bengio, 2009). RBM training methods are usually based on gradient ascent on the Log-Likelihood (LL) of the model parameters given the training data. Since the gradient is intractable, it is often approximated using Gibbs sampling only for a few steps (Hinton et al., 2006; Tieleman, 2008; Tieleman and Hinton, 2009).

Two major problems have been reported when training RBMs. Firstly, the bias of the gradient approximation introduced by using only a few steps of Gibbs sampling may lead to a divergence of the LL during training (Fischer and Igel, 2010a; Schulz et al., 2010). To overcome the divergence problem, Desjardins et al. (2010b) have proposed to use parallel tempering, which is an advanced sampling method that leads to a faster mixing Markov chain and thus to a better approximation of the LL gradient.

Secondly, the learning process is not invariant to the data representation. For example training an RBM on the *MNIST* data set leads to a better model than training it on *1-MNIST* (the data set generated by flipping each bit in *MNIST*). This is due to missing invariance properties of the gradient with respect to these flip transformations and not due to the model's capacity, since an RBM trained on *MNIST* can be transformed in such a way that it models *1-MNIST* with the same LL. Recently, two approaches have been introduced that address the invariance problem. The enhanced gradient (Cho et al., 2011, 2013b) has been designed as an invariant alternative to the true LL gradient of binary RBMs and has been derived by calculating a weighted average over the gradients one gets by applying any possible bit flip combination on the data set. Empirical results suggest that the enhanced gradient leads to more distinct features and thus to better classification results based on the learned hidden representation of the data. Furthermore, in combination with an adaptive learning rate the enhanced gradient leads to more stable training in the sense that good LL values are reached independently of the initial learning rate. Tang and Sutskever (2011), on the other hand have shown empirically that subtracting the data mean from the visible variables leads to a model that can reach similar LL values on the *MNIST* and the *1-MNIST* data set and comparable results to those of the enhanced gradient.[1] Removing the mean from all variables is generally known as the "centering trick" which was originally proposed for feed forward neural networks (LeCun et al., 1998b). It has recently also been applied to the visible and hidden variables of Deep Boltzmann Machines (DBMs, Montavon and Müller, 2012) where it has been shown to lead to an initially better conditioned optimization problem. Furthermore, the learned features

[1] Note, that changing the model such that the mean of the visible variables is removed is not equivalent to removing the mean of the data.

have shown better discriminative properties and centering has improved the generative properties of locally connected DBMs. A related approach applicable to multi-layer perceptrons where the activation functions of the neurons are transformed to have zero mean and zero slope on average was proposed by Raiko et al. (2012). The authors could show that the gradient under this transformation became closer to the natural gradient, which is desirable since the natural gradient follows the direction of steepest ascent in the manifold of probability distributions. Furthermore, the natural gradient is independent of the concrete parameterization of the distributions and is thus clearly the update direction of choice (Amari, 1998). However, it is intractable already for rather small RBMs. Schwehn (2010) and Ollivier et al. (2013) trained binary RBMs and Desjardins et al. (2013) binary DBMs using approximations of the natural gradient obtained by Markov chain Monte Carlo methods. Despite the theoretical arguments for using the natural gradient, the authors concluded that the computational overhead is extreme and it is rather questionable that the natural gradient is efficient for training RBMs or DBMs.

In this work we give a unified view on centering that is applying the centering trick of binary RBMs. We begin with a brief overview over binary RBMs, the standard learning algorithms, the natural gradient of the LL of RBMs, and the basic ideas used to construct the enhanced gradient in Section 8.2. In Section 8.3 we discuss the theoretical properties of centered RBMs, show that centering can be reformulated as a different update rule for normal binary RBMs and that the enhanced gradient is a particular form of centering. Section 8.4 discusses how the parameters of centered and normal binary RBMs should be initialized. Our experimental setup is described in Section 8.5 before we empirically analyze the performance of centered RBMs with different initializations, offset parameters, sampling methods, and learning rates and compare the centered gradient with the natural gradient in Section 8.6. Our work is concluded in Section 8.7.

8.2 Restricted Boltzmann machines

An RBM (Smolensky, 1986) is a bipartite undirected graphical model with a set of m visible variables $\mathbf{V} = (V_1, ..., V_m)$ and n hidden variables $\mathbf{H} = (H_1, ..., H_n)$ taking values $\mathbf{v} = (v_1, ..., v_m)$ and $\mathbf{h} = (h_1, ..., h_n)$, respectively. Since an RBM is a Markov random field, its joint probability distribution is given by a Gibbs distribution

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \mathrm{e}^{-E(\mathbf{v},\mathbf{h})} ,$$

with partition function Z and energy $E(\mathbf{v}, \mathbf{h})$. For binary RBMs, $\mathbf{v} \in \{0,1\}^m$, $\mathbf{h} \in \{0,1\}^n$, and the energy, which defines the bipartite structure, is given by

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T\mathbf{b} - \mathbf{c}^T\mathbf{h} - \mathbf{v}^T\mathbf{W}\mathbf{h} \ ,$$

where the weight matrix $\mathbf{W}$, the visible bias vector $\mathbf{b}$ and the hidden bias vector $\mathbf{c}$ are the parameters of the model, jointly denoted by $\boldsymbol{\theta}$. The partition function which sums over all possible visible and hidden states is given by

$$Z = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \mathrm{e}^{-E(\mathbf{v},\mathbf{h})} \ .$$

RBM training is usually based on gradient ascent using approximations of the LL gradient

$$\nabla\boldsymbol{\theta} = \frac{\partial \langle \log(p(\mathbf{v}|\boldsymbol{\theta})) \rangle_d}{\partial \boldsymbol{\theta}} = -\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_d + \left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_m \ ,$$

where $\langle\cdot\rangle_m$ is the expectation under $p(\mathbf{h}, \mathbf{v})$ and $\langle\cdot\rangle_d$ is the expectation under $p(\mathbf{h}|\mathbf{v})p_e(\mathbf{v})$ with empirical distribution p_e. We use the notation $\nabla\boldsymbol{\theta}$ for the derivative of the LL with respect to $\boldsymbol{\theta}$ in order to be consistent with the notation used by Cho et al. (2011). For binary RBMs the gradient becomes

$$\begin{aligned} \nabla\mathbf{W} &= \langle \mathbf{v}\mathbf{h}^T \rangle_d - \langle \mathbf{v}\mathbf{h}^T \rangle_m \ , \\ \nabla\mathbf{b} &= \langle \mathbf{v} \rangle_d - \langle \mathbf{v} \rangle_m \ , \\ \nabla\mathbf{c} &= \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m \ . \end{aligned}$$

Common RBM training methods approximate $\langle\cdot\rangle_m$ by samples gained by different Markov chain Monte Carlo methods. Sampling k (usually $k = 1$) steps from a Gibbs chain initialized with a data sample yields the Contrastive Divergence (CD, Hinton et al., 2006) algorithm. In stochastic maximum likelihood (Younes, 1991), in the context of RBMs also known as Persistent Contrastive Divergence (PCD, Tieleman, 2008), the chain is not reinitialized with a data sample after parameter updates. This has been reported to lead to better gradient approximations if the learning rate is chosen sufficiently small. Fast Persistent Contrastive Divergence (FPCD, Tieleman and Hinton, 2009) tries to further speed up learning by introducing an additional set of parameters, which is only used for Gibbs sampling during learning. The advanced sampling method Parallel Tempering (PT) introduces additional tempered Gibbs chains corresponding to smoothed versions of $p(\mathbf{v}, \mathbf{h})$. The energy of these distributions is multiplied by $\frac{1}{T}$, where T is referred to as temperature. The higher the temperature of a chain is, the "smoother" the corresponding distribution and the faster the chain mixes. Samples may swap between chains with a probability given by the Metropolis Hastings ratio, which leads to better mixing of the original chain (where $T = 1$). We use PT_c

to denote the RBM training algorithm that uses parallel tempering with c tempered chains as a sampling method. Usually only one step of Gibbs sampling is performed in each tempered chain before allowing samples to swap, and a deterministic even odd algorithm (Lingenheil et al., 2009) is used as a swapping schedule. PT_c increases the mixing rate and has been reported to achieve better gradient approximations than CD and (F)PCD (Desjardins et al., 2010b) with the drawback of having a higher computational cost.

See the introductory paper of Fischer and Igel (2014) for a recent review of RBMs and their training algorithms.

8.2.1 Enhanced gradient

Cho et al. (2011) proposed a different way to update parameters during training of binary RBMs, which is invariant to the data representation.

When transforming the state $(\mathbf{v}, \mathbf{h})$ of a binary RBM by flipping some of its variables (that is $\tilde{v_i} = 1 - v_i$ and $\tilde{h_j} = 1 - h_j$ for some i, j), yielding a new state $(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})$, one can transform the parameters $\boldsymbol{\theta}$ of the RBM to $\tilde{\boldsymbol{\theta}}$ such that $E(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}) = E(\tilde{\mathbf{v}}, \tilde{\mathbf{h}}|\tilde{\boldsymbol{\theta}}) + const$ and thus $p(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}) = p(\tilde{\mathbf{v}}, \tilde{\mathbf{h}}|\tilde{\boldsymbol{\theta}})$ holds. However, if we update the parameters of the transformed model based on the corresponding LL gradient to $\tilde{\boldsymbol{\theta}}' = \tilde{\boldsymbol{\theta}} + \eta \nabla \tilde{\boldsymbol{\theta}}$ and apply the inverse parameter transformation to $\tilde{\boldsymbol{\theta}}'$, the result will differ from $\boldsymbol{\theta}' = \boldsymbol{\theta} + \eta \nabla \boldsymbol{\theta}$. The described procedure of transforming, updating, and transforming back can be regarded as a different way to update $\boldsymbol{\theta}$.

Following this line of thought there exist 2^{n+m} different parameter updates corresponding to the 2^{n+m} possible binary flips of $(\mathbf{v}, \mathbf{h})$. Cho et al. (2011) proposed the enhanced gradient as a weighted sum of these 2^{n+m} parameter updates, which for their choice of weighting is given by

$$\begin{aligned}
\nabla_e \mathbf{W} &= \langle (\mathbf{v} - \langle \mathbf{v} \rangle_d)(\mathbf{h} - \langle \mathbf{h} \rangle_d)^T \rangle_d - \langle (\mathbf{v} - \langle \mathbf{v} \rangle_m)(\mathbf{h} - \langle \mathbf{h} \rangle_m)^T \rangle_m \ , \\
\nabla_e \mathbf{b} &= \langle \mathbf{v} \rangle_d - \langle \mathbf{v} \rangle_m - \nabla_e \mathbf{W} \frac{1}{2} \left(\langle \mathbf{h} \rangle_d + \langle \mathbf{h} \rangle_m \right) \ , \\
\nabla_e \mathbf{c} &= \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m - \nabla_e \mathbf{W}^T \frac{1}{2} \left(\langle \mathbf{v} \rangle_d + \langle \mathbf{v} \rangle_m \right) \ .
\end{aligned}$$

It has been shown that the enhanced gradient is invariant to arbitrary bit flips of the variables and therefore invariant under the data representation, which has been demonstrated on the *MNIST* and *1-MNIST* data set. Furthermore, the authors reported more stable training under various settings in terms of the LL estimate and classification accuracy.

8.2.2 Natural gradient

Following the direction of steepest ascent in the Euclidean parameter space (as given by the standard gradient) does not necessarily correspond to the direction of steepest ascent in the manifold of probability distributions $\{p(\mathbf{v}|\boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}$, which we are actually interested in. To account for the local geometry of the manifold, the Euclidean metric should be replaced by the Fisher information metric defined by $||\boldsymbol{\theta}||_{\mathcal{I}(\boldsymbol{\theta})} = \sqrt{\sum \theta_k \mathcal{I}_{kl}(\boldsymbol{\theta})\, \theta_l}$, where $\mathcal{I}(\boldsymbol{\theta})$ is the Fisher information matrix (Amari, 1998). The kl-th entry of the Fisher information matrix for a parameterized distribution $p(\mathbf{v}|\boldsymbol{\theta})$ is given by

$$\boldsymbol{\mathcal{I}}_{kl}(\theta) = \left\langle \left(\frac{\partial \log\left(p(\mathbf{v}|\boldsymbol{\theta})\right)}{\partial \theta_k} \right) \left(\frac{\partial \log\left(p(\mathbf{v}|\boldsymbol{\theta})\right)}{\partial \theta_l} \right) \right\rangle_m ,$$

where $\langle \cdot \rangle_m$ denotes the expectation under $p(\mathbf{v}|\boldsymbol{\theta})$. The gradient associated with the Fisher metric is called the natural gradient and is given by

$$\nabla_n \boldsymbol{\theta} = \boldsymbol{\mathcal{I}}(\theta)^{-1} \nabla \boldsymbol{\theta} .$$

The natural gradient points in the direction $\delta\boldsymbol{\theta}$ archiving the largest change of the objective function (here the LL) for an infinitesimal small distance $\delta\boldsymbol{\theta}$ between $p(\mathbf{v}|\boldsymbol{\theta})$ and $p(\mathbf{v}|\boldsymbol{\theta} + \boldsymbol{\delta\theta})$ in terms of the Kullback-Leibler divergence (Amari, 1998). This makes the natural gradient independent of the parameterization including the invariance to flips of the data as a special case. Thus, the natural gradient is clearly the update direction of choice.

For binary RBMs the entries of the Fisher information matrix (Amari et al., 1992; Desjardins et al., 2013; Ollivier et al., 2013) are given by

$$\begin{aligned}
\boldsymbol{\mathcal{I}}_{w_{ij},w_{uv}}(\theta) = \boldsymbol{\mathcal{I}}_{,w_{uv},w_{ij}}(\theta) &= \langle v_i h_j v_u h_v \rangle_m - \langle v_u h_v \rangle_m \langle v_u h_v \rangle_m \\
&= Cov_m(v_i h_j, v_u h_v) , \\
\boldsymbol{\mathcal{I}}_{w_{ij},b_u}(\theta) = \boldsymbol{\mathcal{I}}_{b_u,w_{ij}}(\theta) &= Cov_m(v_i h_j, v_u) , \\
\boldsymbol{\mathcal{I}}_{w_{ij},c_v}(\theta) = \boldsymbol{\mathcal{I}}_{c_v,w_{ij}}(\theta) &= Cov_m(v_i h_j, h_v) , \\
\boldsymbol{\mathcal{I}}_{b_i,b_u}(\theta) = \boldsymbol{\mathcal{I}}_{b_u,b_i}(\theta) &= Cov_m(v_i, v_u) , \\
\boldsymbol{\mathcal{I}}_{c_j,c_v}(\theta) = \boldsymbol{\mathcal{I}}_{c_v,c_j}(\theta) &= Cov_m(h_j, h_v) .
\end{aligned}$$

Since these expressions involve expectations under the model distribution they are not tractable in general, but can be approximated using MCMC methods (Ollivier et al., 2013; Desjardins et al., 2013). Furthermore, a diagonal approximation of the Fisher information matrix could be used. However, the approximation of the natural gradient is still computationally very expensive so that the practical usability remains questionable (Desjardins et al., 2013).

8.3 Centered restricted Boltzmann machines

Inspired by the centering trick proposed by LeCun et al. (1998b), Tang and Sutskever (2011) have addressed the flip-invariance problem by changing the energy of the RBM in a way that the mean of the input data is removed. Montavon and Müller (2012) have extended the idea of centering to the visible and hidden variables of DBMs and have shown that centering improves the conditioning of the underlying optimization problem, leading to models with better discriminative properties for DBMs in general and better generative properties in the case of locally connected DBMs.

Following their line of thought, the energy for a centered binary RBM where the visible and hidden variables are shifted by the offset parameters $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_m)$ and $\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_n)$, respectively, can be formulated as

$$E(\mathbf{v}, \mathbf{h}) = -(\mathbf{v} - \boldsymbol{\mu})^T \mathbf{b} - \mathbf{c}^T (\mathbf{h} - \boldsymbol{\lambda}) - (\mathbf{v} - \boldsymbol{\mu})^T \mathbf{W} (\mathbf{h} - \boldsymbol{\lambda}) \ . \tag{8.1}$$

By setting both offsets to zero one retains the normal binary RBM. Setting $\boldsymbol{\mu} = \langle \mathbf{v} \rangle_d$ and $\boldsymbol{\lambda} = \mathbf{0}$ leads to the model introduced by Tang and Sutskever (2011), and by setting $\boldsymbol{\mu} = \langle \mathbf{v} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_d$ we get a shallow variant of the centered DBM analyzed by Montavon and Müller (2012).

The conditional probabilities for a variable taking the value one are given by

$$p(V_i = 1|\mathbf{h}) = \sigma(\mathbf{w_{i*}} (\mathbf{h} - \boldsymbol{\lambda}) + b_i) \ , \tag{8.2}$$

$$p(H_j = 1|\mathbf{v}) = \sigma((\mathbf{v} - \boldsymbol{\mu})^T \mathbf{w_{*j}} + c_j) \ , \tag{8.3}$$

where $\sigma(\cdot)$ is the sigmoid function, $\mathbf{w_{i*}}$ represents the ith row, and $\mathbf{w_{*j}}$ the jth column of the weight matrix $\mathbf{W}$.

The LL gradient now takes the form

$$\nabla \mathbf{W} = \langle (\mathbf{v} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_d - \langle (\mathbf{v} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_m \ , \tag{8.4}$$

$$\nabla \mathbf{b} = \langle \mathbf{v} - \boldsymbol{\mu} \rangle_d - \langle \mathbf{v} - \boldsymbol{\mu} \rangle_m = \langle \mathbf{v} \rangle_d - \langle \mathbf{v} \rangle_m \ , \tag{8.5}$$

$$\nabla \mathbf{c} = \langle \mathbf{h} - \boldsymbol{\lambda} \rangle_d - \langle \mathbf{h} - \boldsymbol{\lambda} \rangle_m = \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m \ . \tag{8.6}$$

$\nabla \mathbf{b}$ and $\nabla \mathbf{c}$ are independent of the choice of $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ and thus centering only affects $\nabla \mathbf{W}$. It can be shown (see the appendix in Section 8.8) that the gradient of a centered RBM is invariant to flip transformations if a flip of v_i to $1 - v_i$ implies a change of μ_i to $1 - \mu_i$, and a flip of h_j to $1 - h_j$ implies a change of λ_j to $1 - \lambda_j$. This obviously holds for $\mu_i = 0.5$ and $\lambda_j = 0.5$ but also for any expectation value over v_i and h_j under any distribution. Note, that the invariance property also generalizes to DBMs.

If we set $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ to the expectation values of the variables, these values may depend on the RBM parameters (think for example about $\langle \mathbf{h} \rangle_d$) and thus they might change during training. Consequently, a learning algorithm for centered RBM needs

to update the offset values to match the expectations under the distribution that has changed with a parameter update. When updating the offsets one needs to transform the RBM parameters such that the modeled probability distribution stays the same. An RBM with offsets $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ can be transformed to an RBM with offsets $\boldsymbol{\mu}'$ and $\boldsymbol{\lambda}'$ by

$$\begin{aligned} \mathbf{W}' &= \mathbf{W} \ , & (8.7) \\ \mathbf{b}' &= \mathbf{b} + \mathbf{W}\left(\boldsymbol{\lambda}' - \boldsymbol{\lambda}\right) \ , & (8.8) \\ \mathbf{c}' &= \mathbf{c} + \mathbf{W}^T\left(\boldsymbol{\mu}' - \boldsymbol{\mu}\right) \ , & (8.9) \end{aligned}$$

such that $E(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = E(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}', \boldsymbol{\mu}', \boldsymbol{\lambda}') + const$, is guaranteed. Obviously, this can be used to transform a centered RBM to a normal RBM and vice versa, highlighting that centered and normal RBMs are just different parametrizations of the same model class.

If the intractable model mean is used for the offsets, they have to be approximated by samples. Furthermore, when $\boldsymbol{\lambda}$ is chosen to be $\langle \mathbf{h} \rangle_d$ or $\langle \mathbf{h} \rangle_m$ or when $\boldsymbol{\mu}$ is chosen to be $\langle \mathbf{v} \rangle_m$ one could either approximate the mean values using the sampled states or the corresponding conditional probabilities. But due to the Rao-Blackwell theorem an estimation based on the probabilities has lower variance and therefore is the approximation of choice.

Algorithm 8.1 shows pseudo code for training a centered binary RBM, where we use $\langle \cdot \rangle$ to denote the average over samples from the current batch. Thus, for example we write $\langle \mathbf{v}_d \rangle$ for the average value of data samples $\mathbf{v}_d$ in the current batch, which is used as an approximation for the expectation of $\mathbf{v}$ under the data distribution, that is $\langle \mathbf{v} \rangle_d$. Similarly, $\langle \mathbf{h}_d \rangle$ approximates $\langle \mathbf{h} \rangle_d$ using the hidden samples $\mathbf{h}_d$ in the current batch.

Note that in Algorithm 8.1 the update of the offsets is performed before the gradient is calculated. This is in contrast to the algorithm for centered DBMs proposed by Montavon and Müller (2012), where the update of the offsets and the reparameterization follows after the gradient update (that is, the estimates of the offsets in one learning iteration are based on samples gained from the model of the previous iteration). However, the proposed DBM algorithm smooths the offset estimations by an exponentially moving average over the sample means from many iterations, so that the choice of the sample set used for the offset estimation should be less relevant. In Algorithm 8.1 an exponentially moving average is obtained if the sliding factor ν is set to $0 < \nu < 1$ and prevented if $\nu = 1$. The effects of using an exponentially moving average are empirically analyzed in Section 8.6.2.

Algorithm 8.1: Training centered RBMs

Initialize $\mathbf{W}$ /* *i.e.* $\mathbf{W} \leftarrow \mathcal{N}(0, 0.01)^{N \times M}$ */
Initialize $\boldsymbol{\mu}, \boldsymbol{\lambda}$ /* *i.e.* $\boldsymbol{\mu} \leftarrow \langle \texttt{data} \rangle, \boldsymbol{\lambda} \leftarrow \mathbf{0.5}$ */
Initialize $\mathbf{b}, \mathbf{c}$ /* *i.e.* $\mathbf{b} \leftarrow \sigma^{-1}(\boldsymbol{\mu}), \mathbf{c} \leftarrow \sigma^{-1}(\boldsymbol{\lambda})$ */
Initialize $\eta, \nu_\mu, \nu_\lambda$ /* *i.e.* $\eta, \nu_\mu, \nu_\lambda \in \{0.001, ..., 0.1\}$ */
repeat
  foreach batch **in** data **do**
    foreach *sample* $\mathbf{v}_d$ **in** batch **do**
      Calculate $\mathbf{h}_d = p(H_j = 1 | \mathbf{v}_d)$ /* Eq. (8.3) */
      Sample $\mathbf{v}_m$ *from RBM* /* Eqs. (8.2), (8.3) */
      Calculate $\mathbf{h}_m = p(H_j = 1 | \mathbf{v}_m)$ /* Eq. (8.3) */
      Store $\mathbf{v}_m, \mathbf{h}_d, \mathbf{h}_m$
    Estimate $\boldsymbol{\mu}_{new}$ /* *i.e.* $\boldsymbol{\mu}_{new} \leftarrow \langle \mathbf{v}_d \rangle$ */
    Estimate $\boldsymbol{\lambda}_{new}$ /* *i.e.* $\boldsymbol{\lambda}_{new} \leftarrow \langle \mathbf{h}_d \rangle$ */
    /* Transform parameters with respect to the new offsets */
    $\mathbf{b} \leftarrow \mathbf{b} + \nu_\lambda \mathbf{W} (\boldsymbol{\lambda}_{new} - \boldsymbol{\lambda})$ /* Eq. (8.8) */
    $\mathbf{c} \leftarrow \mathbf{c} + \nu_\mu \mathbf{W}^T (\boldsymbol{\mu}_{new} - \boldsymbol{\mu})$ /* Eq. (8.9) */
    /* Update offsets using exp. moving averages with sliding factors ν_μ and ν_λ */
    $\boldsymbol{\mu} \leftarrow (1 - \nu_\mu)\boldsymbol{\mu} + \nu_\mu \boldsymbol{\mu}_{new}$
    $\boldsymbol{\lambda} \leftarrow (1 - \nu_\lambda)\boldsymbol{\lambda} + \nu_\lambda \boldsymbol{\lambda}_{new}$
    /* Update parameters using gradient ascent with learning rate η */
    $\nabla \mathbf{W} \leftarrow \langle (\mathbf{v}_d - \boldsymbol{\mu})(\mathbf{h}_d - \boldsymbol{\lambda})^T \rangle - \langle (\mathbf{v}_m - \boldsymbol{\mu})(\mathbf{h}_m - \boldsymbol{\lambda})^T \rangle$ /* Eq. (8.4) */
    $\nabla \mathbf{b} \leftarrow \langle \mathbf{v}_d \rangle - \langle \mathbf{v}_m \rangle$ /* Eq. (8.5) */
    $\nabla \mathbf{c} \leftarrow \langle \mathbf{h}_d \rangle - \langle \mathbf{h}_m \rangle$ /* Eq. (8.6) */
    $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla \mathbf{W}$
    $\mathbf{b} \leftarrow \mathbf{b} + \eta \nabla \mathbf{b}$
    $\mathbf{c} \leftarrow \mathbf{c} + \eta \nabla \mathbf{c}$
until *stopping criteria is met*
/* Transform network to a normal binary RBM if desired */
$\mathbf{b} \leftarrow \mathbf{b} - \mathbf{W}\boldsymbol{\lambda}$ /* Eq. (8.8) */
$\mathbf{c} \leftarrow \mathbf{c} - \mathbf{W}^T \boldsymbol{\mu}$ /* Eq. (8.9) */
$\boldsymbol{\mu} \leftarrow \mathbf{0}$
$\boldsymbol{\lambda} \leftarrow \mathbf{0}$

8.3.1 Centered Gradient

We now use the centering trick to derive a centered parameter update, which can replace the gradient during the training of normal binary RBMs. Similar to the derivation of the enhanced gradient we can transform a normal binary to a centered RBM, perform a gradient update, and transform the RBM back (see the appendix in Section 8.8 for the derivation). This yields the following parameter updates, which we refer to as centered gradient

$$\nabla_c \mathbf{W} = \langle (\mathbf{v} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_d - \langle (\mathbf{v} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_m \ , \tag{8.10}$$

$$\nabla_c \mathbf{b} = \langle \mathbf{v} \rangle_d - \langle \mathbf{v} \rangle_m - \nabla_c \mathbf{W} \boldsymbol{\lambda} \ , \tag{8.11}$$

$$\nabla_c \mathbf{c} = \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m - \nabla_c \mathbf{W}^T \boldsymbol{\mu} \ . \tag{8.12}$$

Notice that by setting $\boldsymbol{\mu} = \frac{1}{2} \left(\langle \mathbf{v} \rangle_d + \langle \mathbf{v} \rangle_m \right)$ and $\boldsymbol{\lambda} = \frac{1}{2} \left(\langle \mathbf{h} \rangle_d + \langle \mathbf{h} \rangle_m \right)$ the centered gradient becomes equal to the enhanced gradient. Thus, it becomes clear that the enhanced gradient is a special case of centering. This can also be concluded from the derivation of the enhanced gradient for Gaussian visible variables by Cho et al. (2013a).

The enhanced gradient has been designed such, that the weight updates become the difference of the covariances between one visible and one hidden variable under the data and the model distribution. Interestingly, one gets the same weight update for two other choices of offset parameters: either $\boldsymbol{\mu} = \langle \mathbf{v} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_m$ or $\boldsymbol{\mu} = \langle \mathbf{v} \rangle_m$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_d$. However, these offsets result in different update rules for the bias parameters.

Algorithm 8.2 shows pseudo code for training a normal binary RBM using the centered gradient, which is equivalent to training a centered binary RBM using Algorithm 8.1. Both algorithms can easily be extended to DBMs and Boltzmann machines with other types of units.

8.4 Initialization of the model parameters

It is a common way to initialize the weight matrix to small random values to break the symmetry. The bias parameters are often initialized to zero. However, there exists a more reasonable initialization for the bias parameters.

Hinton (2012) proposed to initialize the visible bias parameter b_i to $\ln(p_i/(1-p_i))$, where p_i is the proportion of the data points in which unit i is on (that is $p_i = \langle v_i \rangle_d$). He states that if this is not done, the hidden units are used to activate the ith visible unit with a probability of approximately p_i in the early stage of training.

We argue that this initialization is in fact reasonable since it corresponds to the Maximum Likelihood Estimate (MLE) of the visible bias given the data for an RBM with zero weight matrix, given by

Algorithm 8.2: Training RBMs using the centered gradient

Initialize $\mathbf{W}$ /* *i.e.* $\mathbf{W} \leftarrow \mathcal{N}(0, 0.01)^{N \times M}$ */
Initialize $\boldsymbol{\mu}, \boldsymbol{\lambda}$ /* *i.e.* $\boldsymbol{\mu} \leftarrow \langle \texttt{data} \rangle, \boldsymbol{\lambda} \leftarrow \mathbf{0.5}$ */
Initialize $\mathbf{b}, \mathbf{c}$ /* *i.e.* $\mathbf{b} \leftarrow \sigma^{-1}(\boldsymbol{\mu}), \mathbf{c} \leftarrow \sigma^{-1}(\boldsymbol{\lambda})$ */
Initialize $\eta, \nu_\mu, \nu_\lambda$ /* *i.e.* $\eta, \nu_\mu, \nu_\lambda \in \{0.001, ..., 0.1\}$ */
repeat
  foreach batch **in** data **do**
    foreach ample $\mathbf{v}_d$ **in** batch **do**
      Calculate $\mathbf{h}_d = p(H_j = 1 | \mathbf{v}_d)$ /* Eq. (8.3) */
      Sample $\mathbf{v}_m$ *from RBM* /* Eqs. (8.2), (8.3) */
      Calculate $\mathbf{h}_m = p(H_j = 1 | \mathbf{v}_m)$ /* Eq. (8.3) */
      Store $\mathbf{v}_m, \mathbf{h}_d, \mathbf{h}_m$
    Estimate $\boldsymbol{\mu}_{new}$ /* *i.e.* $\boldsymbol{\mu}_{new} \leftarrow \langle \mathbf{v}_d \rangle$ */
    Estimate $\boldsymbol{\lambda}_{new}$ /* *i.e.* $\boldsymbol{\lambda}_{new} \leftarrow \langle \mathbf{h}_d \rangle$ */
    /* Update offsets using exp. moving averages with sliding factors ν_μ and ν_λ */
    $\boldsymbol{\mu} \leftarrow (1 - \nu_\mu)\boldsymbol{\mu} + \nu_\mu \boldsymbol{\mu}_{new}$
    $\boldsymbol{\lambda} \leftarrow (1 - \nu_\lambda)\boldsymbol{\lambda} + \nu_\lambda \boldsymbol{\lambda}_{new}$
    /* Update parameters using the centered gradient with learning rate η */
    $\nabla_c \mathbf{W} \leftarrow \langle (\mathbf{v}_d - \boldsymbol{\mu})(\mathbf{h}_d - \boldsymbol{\lambda})^T \rangle - \langle (\mathbf{v}_m - \boldsymbol{\mu})(\mathbf{h}_m - \boldsymbol{\lambda})^T \rangle$ /* Eq. (8.10) */
    $\nabla_c \mathbf{b} \leftarrow \langle \mathbf{v}_d \rangle - \langle \mathbf{v}_m \rangle - \nabla_c \mathbf{W} \boldsymbol{\lambda}$ /* Eq. (8.11) */
    $\nabla_c \mathbf{c} \leftarrow \langle \mathbf{h}_d \rangle - \langle \mathbf{h}_m \rangle - \nabla_c \mathbf{W}^T \boldsymbol{\mu}$ /* Eq. (8.12) */
    $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla_c \mathbf{W}$
    $\mathbf{b} \leftarrow \mathbf{b} + \eta \nabla_c \mathbf{b}$
    $\mathbf{c} \leftarrow \mathbf{c} + \eta \nabla_c \mathbf{c}$
until *stopping criteria is met*

$$\mathbf{b}^* = \ln\left(\frac{\langle \mathbf{v}\rangle_d}{\mathbf{1} - \langle \mathbf{v}\rangle_d}\right) = -\ln\left(\frac{\mathbf{1}}{\langle \mathbf{v}\rangle_d} - \mathbf{1}\right) = \sigma^{-1}(\langle \mathbf{v}\rangle_d) \ , \tag{8.13}$$

where σ^{-1} is the inverse sigmoid function. Notice that the MLE of the visible bias for an RBM with zero weights is the same whether the RBM is centered or not. The conditional probability of the visible variables (8.2) of an RBM with this initialization is then given by $p(\mathbf{v} = \mathbf{1}|\mathbf{h}) = \sigma(\sigma^{-1}(\langle \mathbf{v}\rangle_d)) = \langle \mathbf{v}\rangle_d$, where $p(\mathbf{v} = \mathbf{1}|\mathbf{h})$ denotes the vector containing the elements $p(v_i = 1|\mathbf{h})$, $i = 1, \ldots, m$. Thus the mean of the data is initially modeled only by the bias values and the weights are free to model higher order statistics in the beginning of training. For the unknown hidden variables it is reasonable to assume an initial mean of 0.5 so that the MLE of the hidden bias for an RBM with zero weights is given by $\mathbf{c}^* = \sigma^{-1}(0.5) = 0.0$. These considerations still hold approximately if the weights are not zero but initialized to small random values.

Montavon and Müller (2012) suggested to initialize the bias parameters to the inverse sigmoid of the initial offset parameters. They argue that this initialization leads to a good starting point, because it guarantees that the Boltzmann machine is initially centered. Actually, if the initial offsets are set to $\mu_i = \langle v_i\rangle_d$ and $\lambda_j = 0.5$ the initialization suggested by Montavon and Müller (2012) is equal to the initialization to the MLEs as follows from equation (8.13).

8.5 Methods

As shown in the previous section the algorithms described by Montavon and Müller (2012), Tang and Sutskever (2011), and Cho et al. (2011) can all be viewed as different ways of applying the centering trick. They differ in the choice of the offset parameters and in the way of approximating them, either based on the samples gained from the model in the previous learning step or from the current one, using an exponentially moving average or not. The question arises, how RBMs should be centered to achieve the best performance in terms of the LL. In the following we analyze the different ways of centering empirically and try to derive a deeper understanding of why centering is beneficial.

For simplicity we introduce the following shorthand notation. We use d to denote the data mean $\langle\cdot\rangle_d$, m for the model mean $\langle\cdot\rangle_m$, a for the average of the means $\frac{1}{2}\langle\cdot\rangle_d + \frac{1}{2}\langle\cdot\rangle_m$, and 0 if the offsets is set to zero. We indicate the choice of $\boldsymbol{\mu}$ in the first and the choice of $\boldsymbol{\lambda}$ in the second place, for example dm translates to $\boldsymbol{\mu} = \langle \mathbf{v}\rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h}\rangle_m$. We add a superscribed b or a to denote whether the reparameterization is performed before or after the gradient update. If a sliding factor smaller than one and thus an exponentially moving average is used a subscript s is added. Thus, we indicate the variant of Montavon and Müller (2012) by dd_s^a, the one of Cho et al. (2011) by aa^b, the

data normalization of Tang and Sutskever (2011) by *d0*, and the normal binary RBM simply by *00*.

We begin our analysis with RBMs, where one layer is small enough to guarantee that the exact LL is still tractable. In a first set of experiments we analyze the four algorithms described above in terms of the evolution of the LL during training. In a second set of experiments we analyze the effect of the initialization described in Section 8.4. We proceed with a comparison of the effects of estimating offset values and reparameterizing the parameters before or after the gradient update. Afterwards we analyze the effects of using an exponentially moving average to approximate the offset values in the different algorithms and of choosing other offset values. To verify whether the results scale to more realistic problem sizes we compare the algorithms on *MNIST* using 500 hidden units. Finally, we compare the normal and the centered gradient with the natural gradient.

8.5.1 Benchmark problems

For our analysis we consider three different benchmark problems.

The ***Bars & Stripes*** (MacKay, 2002) problem consists of quadratic patterns of size $D \times D$ that can be generated as follows. First, a vertical or horizontal orientation is chosen randomly with equal probability. Then the state of all pixels of every row or column is chosen uniformly at random. This leads to $N = 2^{D+1}$ patterns where the completely uniform patterns occur twice as often as the others. The data set is symmetric in terms of the amount of zeros and ones and thus the flipped and unflipped problems are equivalent. An upper bound of the LL is given by $(N-4)\ln\left(\frac{1}{N}\right) + 4\ln\left(\frac{2}{N}\right)$. For our experiments we used $D = 3$ or $D = 2$ (only in Section 8.6.7) leading to an upper bound of -41.59 and -13.86, respectively.

The ***Shifting Bar*** data set is an artificial benchmark problem we have designed to be asymmetric in terms of the amount of zeros and ones in the data. For an input dimensionality N, a bar of length $0 < B < N$ has to be chosen, where $\frac{B}{N}$ expresses the percentage of ones in the data set. A position $0 \leq p < N$ is chosen uniformly at random and the states of the following B pixels are set to one, where a wrap around is used if $p + B \geq N$. The states of the remaining pixels are set to zero. This leads to N different patterns with equal probability and an upper bound of the LL of $N\ln\left(\frac{1}{N}\right)$. For our experiments we used $N = 9$, $B = 1$ and its flipped version ***Flipped Shifting Bar***, which we get for $N = 9$, $B = 8$, both having an upper LL bound of -19.78.

The ***MNIST*** (LeCun et al., 1998b) database of handwritten digits has become a standard benchmark problem for RBMs. It consists of $60,000$ training and $10,000$ testing examples of gray value handwritten digits of size 28×28. After binarization (with a threshold of 0.5) the data set contains 13.3% ones, similar to the *Shifting Bar*

problem, which for our choice of N and B contains 11.1% ones. We refer to the data set where each bit of MNIST is flipped (that is each one is replaced by a zero and *vice versa*) as ***1-MNIST***. To our knowledge, the best reported performance in terms of the average LL per sample of an RBM with 500 hidden units on *MNIST* test data is -84 (Salakhutdinov, 2008; Salakhutdinov and Murray, 2008; Tang and Sutskever, 2011; Cho et al., 2013b).

8.5.2 Experimental setup

The RBMs weight matrices were initialized with random values sampled from a Gaussian with zero mean and a standard deviation of 0.01. If not stated otherwise the visible and hidden biases, and offsets were initialized as described in Section 8.4. We used CD and PCD with k steps of Gibbs sampling (CD-k, PCD-k) and PT$_c$ for training, where the c temperatures were distributed uniformly form 0 to 1. All experiments where repeated 25 times. We used 4 hidden units when modeling *Bars & Stripes* and *Shifting Bar*. For these data sets batch training was performed for $50,000$ gradient updates, where the LL was evaluated every 50th gradient update. We used either 16 or 500 hidden units together with mini-batch training with a batch size of 100 when modeling *MNIST*. In the case of 16 hidden units the model was trained for 100 epochs, each consisting of 600 gradient updates and the exact LL was evaluated after each epoch. In the case of 500 hidden units the model was trained for 200 epochs, each consisting of 600 gradient updates and the LL was estimated every 10th epoch using Annealed Importance Sampling (AIS), where we used the same setup as described by Salakhutdinov and Murray (2008).

8.6 Results

All tables given in this section show the average maximum LL and the corresponding standard deviation reached during training with different learning algorithms over the 25 trials. In some cases the final average LL reached at the end of training is given in parenthesis to indicate a potential divergence of the LL. For reasons of readability, the average LL was divided by the number of training samples in the case of *MNIST*. In order to check if the result of the best method within one row differs significantly from the others we performed pairwise signed Wilcoxon rank-sum tests (with $p = 0.05$). The best results are highlighted in bold. This can be more than one value if the significance test between these values was negative.

Algorithm-η	aa^b	dd^a_s	$d0$	00
Bars & Stripes				
CD-1-0.1	**-60.85** ±1.91 (-69.1)	**-60.41** ±2.08 (-68.8)	**-60.88** ±3.95 (-70.9)	-65.05 ±3.60 (-78.1)
CD-1-0.05	**-60.37** ±1.87 (-65.0)	**-60.25** ±2.13 (-64.2)	**-60.74** ±3.57 (-65.1)	-64.99 ±3.63 (-71.2)
CD-1-0.01	**-61.00** ±1.54 (-61.1)	-61.22 ±1.49 (-61.3)	-63.28 ±3.01 (-63.3)	-68.41 ±2.91 (-68.6)
PCD-1-0.1	-55.65 ±0.86 (-360.6)	**-54.75** ±1.46 (-91.2)	-56.65 ±3.88 (-97.3)	-57.27 ±4.69 (-84.3)
PCD-1-0.05	-54.29 ±1.25 (-167.4)	**-53.60** ±1.48 (-67.2)	-56.50 ±5.26 (-72.5)	-58.16 ±5.50 (-70.6)
PCD-1-0.01	**-54.26** ±0.79 (-55.3)	-56.68 ±0.73 (-56.8)	-60.83 ±3.76 (-61.0)	-64.52 ±2.94 (-64.6)
PT_{10}-0.1	-52.55 ±3.43 (-202.5)	**-51.13** ±0.85 (-52.1)	-55.37 ±5.44 (-56.7)	-53.99 ±3.73 (-55.3)
PT_{10}-0.05	**-51.84** ±0.98 (-70.7)	**-51.87** ±1.05 (-52.3)	-56.11 ±5.79 (-56.6)	-56.06 ±4.50 (-56.8)
PT_{10}-0.01	**-53.36** ±1.26 (-53.8)	-56.73 ±0.77 (-56.8)	-61.24 ±4.58 (-61.3)	-64.70 ±3.53 (-64.7)
MNIST				
CD-1-0.1	-152.6 ±0.89 (-158.5)	**-150.9** ±1.53 (-154.6)	**-151.3** ±1.77 (-154.8)	-165.9 ±1.90 (-168.4)
CD-1-0.05	-152.5 ±1.14 (-156.1)	**-151.2** ±1.89 (-154.3)	**-151.6** ±1.90 (-154.6)	-167.7 ±1.66 (-169.0)
CD-1-0.01	**-153.0** ±1.10 (-153.2)	**-152.4** ±1.81 (-152.8)	**-153.5** ±2.30 (-154.0)	-171.3 ±1.49 (-172.4)
PCD-1-0.1	-147.5 ±1.09 (-177.6)	**-140.9** ±0.61 (-145.2)	-142.9 ±0.74 (-147.2)	-160.7 ±4.87 (-169.4)
PCD-1-0.05	-145.3 ±0.61 (-162.4)	**-140.0** ±0.45 (-142.8)	-141.1 ±0.65 (-143.6)	-173.4 ±4.42 (-178.1)
PCD-1-0.01	-143.0 ±0.29 (-144.7)	**-140.7** ±0.42 (-141.4)	-141.7 ±0.49 (-142.5)	-198.0 ±4.78 (-198.4)
PT_{10}-0.01	**-247.1** ±12.52 (-643.4)	**-141.5** ±0.54 (-143.6)	-144.0 ±0.61 (-147.6)	-148.8 ±1.15 (-153.6)

Table 8.1: Average maximum LL on (top) the *Bars & Stripes* data set and (bottom) the *MNIST* data set using different sampling methods and learning rates η.

Algorithm-η	aa^b	dd^a_s	$d0$	00
Shifting Bar				
CD-1-0.2	**-20.52** ±1.09 (-21.9)	**-20.32** ±0.74 (-20.6)	-21.72 ±1.21 (-22.5)	-21.89 ±1.42 (-22.6)
CD-1-0.1	**-20.97** ±1.14 (-21.5)	**-20.79** ±0.86 (-20.9)	**-21.19** ±0.82 (-21.4)	-21.40 ±0.88 (-21.6)
CD-1-0.05	**-21.11** ±0.78 (-21.2)	-22.72 ±0.67 (-22.7)	-26.89 ±0.29 (-26.9)	-26.11 ±0.40 (-26.1)
PCD-1-0.2	-21.71 ±0.81 (-237.2)	**-21.02** ±0.52 (-32.4)	-21.62 ±0.66 (-31.9)	-21.86 ±0.75 (-31.7)
PCD-1-0.1	-21.10 ±0.59 (-87.4)	**-20.92** ±0.73 (-23.3)	-21.74 ±0.76 (-23.7)	-21.52 ±0.89 (-23.3)
PCD-1-0.05	**-20.96** ±0.70 (-26.0)	-22.48 ±0.60 (-22.6)	-26.83 ±0.36 (-26.8)	-26.04 ±0.48 (-26.1)
PT_{10}-0.2	-20.87 ±0.86 (-31.9)	**-20.38** ±0.77 (-20.9)	-21.14 ±1.15 (-21.6)	-21.82 ±1.22 (-22.3)
PT_{10}-0.1	**-20.57** ±0.60 (-21.5)	**-20.51** ±0.58 (-20.7)	-21.22 ±0.91 (-21.4)	-21.06 ±0.92 (-21.2)
PT_{10}-0.05	**-20.69** ±0.89 (-20.8)	-22.39 ±0.68 (-22.4)	-26.94 ±0.30 (-27.0)	-26.17 ±0.38 (-26.2)
Flipped Shifting Bar				
CD-1-0.2	**-20.39** ±0.86 (-21.3)	**-20.42** ±0.80 (-20.8)	-21.55 ±1.33 (-22.3)	-27.98 ±0.26 (-28.2)
CD-1-0.1	**-20.57** ±0.83 (-20.9)	-20.85 ±0.82 (-21.0)	-21.04 ±0.75 (-21.2)	-28.28 ±0.00 (-28.4)
CD-1-0.05	**-21.11** ±0.77 (-21.2)	-22.63 ±0.66 (-22.6)	-26.85 ±0.34 (-26.9)	-28.28 ±0.00 (-28.3)
PCD-1-0.2	-21.56 ±0.57 (-310.8)	**-20.97** ±0.65 (-32.3)	-21.89 ±0.86 (-32.6)	-28.01 ±0.26 (-28.3)
PCD-1-0.1	-21.17 ±0.60 (-88.3)	**-20.72** ±0.50 (-23.1)	-21.28 ±0.71 (-23.2)	-28.28 ±0.00 (-28.4)
PCD-1-0.05	**-21.01** ±0.77 (-25.6)	-22.30 ±0.64 (-22.4)	-26.90 ±0.34 (-26.9)	-28.28 ±0.00 (-28.3)
PT_{10}-0.2	-20.60 ±0.66 (-33.2)	**-20.25** ±0.55 (-20.7)	-20.79 ±0.87 (-21.4)	-28.01 ±0.27 (-28.2)
PT_{10}-0.1	**-20.78** ±0.82 (-21.6)	**-20.68** ±0.69 (-20.8)	-21.11 ±0.67 (-21.3)	-28.28 ±0.00 (-28.4)
PT_{10}-0.05	**-20.90** ±0.85 (-21.1)	-22.39 ±0.65 (-22.4)	-26.87 ±0.35 (-26.9)	-28.28 ±0.00 (-28.3)

Table 8.2: Average maximum LL on (top) the *Shifting Bar* data set and (bottom) the *Flipped Shifting Bar* data set using different sampling methods and learning rates.

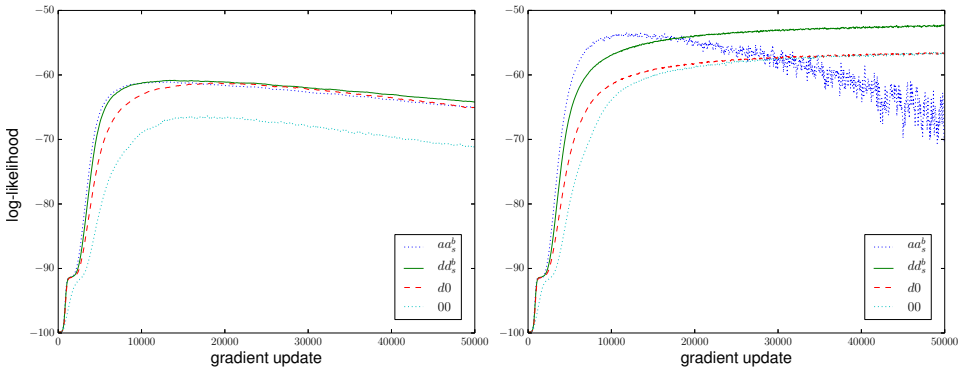


Figure 8.1: Average LL during training on the *Bars & Stripes* data set for the standard centering methods. (left) When CD-1 is used for sampling and the learning rate is $\eta = 0.05$ and (right) when PT_{10} is used for sampling and $\eta = 0.05$.

8.6.1 Comparison of the standard methods

The comparison of the learning performance of the previously described algorithms dd^a_s, aa^b, *d0*, and *00* (using their originally proposed initializations) shows that training a centered RBM leads to significantly higher LL values than training a normal binary RBM (see Table 8.1 for the results for *Bars & Stripes* and *MNIST* and Table 8.2 for the results for *Shifting Bar* and *Flipped Shifting Bar*). Figure 8.1 illustrates on the *Bars & Stripes* data set that centering both the visible and the hidden variables (dd^a_s and aa^b) compared to centering only the visible variables (*d0*) accelerates the learning and leads to a higher LL when using PT. The same holds for PCD as can be seen from Table 8.1. Thus centered RBMs can form more accurate models of the data distribution than normal RBMs. This is in contrast to the observations made for DBMs by Montavon and Müller (2012), which found a better generative performance of centering only in the case of locally connected DBMs.

It can also be seen from Figure 8.1 that all methods show divergence in combination with CD (as described before by Fischer and Igel (2010a) for normal RBMs), which can be prevented for dd^a_s, *d0*, and *00* when using PT. This can be explained by the fact that PT leads to faster mixing Markov chains and thus less biased gradient approximations. The *aa* algorithm however suffers from severe divergence of the LL when PT is used, which is even worse than with CD. This divergence problem arises for all learning rates as indicated by the LL values reached at the end of training (given in parentheses) in Table 8.1 and Table 8.2. The divergence occurs the earlier and faster the bigger the learning rate, while for the other algorithms we never observed

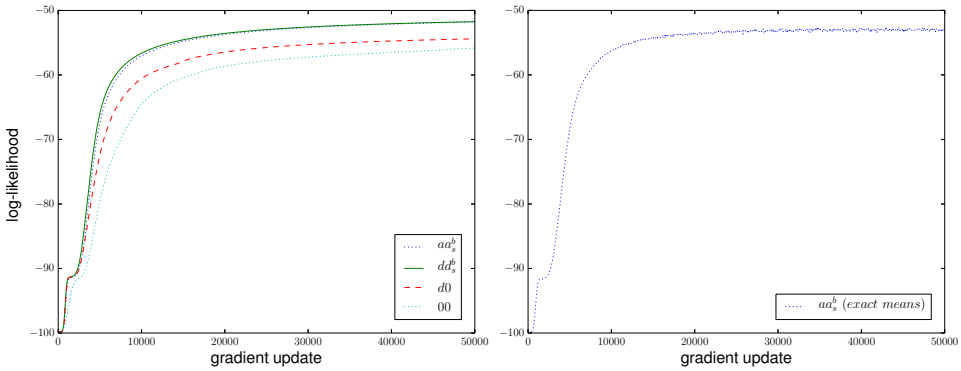


Figure 8.2: Average LL during training on the *Bars & Stripes* data set for the standard centering methods. (left) When the exact gradient is used, with approximated offsets and (right) when PT_{10} is used for estimating the gradient while the mean values for the offsets are calculated exactly. In both cases a learning rate of $\eta = 0.05$ was used.

divergence in combination with PT even for very big learning rates and long training time.

These observations raise the question whether the divergence problem of the enhanced gradient is induced by setting the offsets to $0.5(\langle \mathbf{v} \rangle_d + \langle \mathbf{v} \rangle_m)$ and $0.5(\langle \mathbf{h} \rangle_d + \langle \mathbf{h} \rangle_m)$ or by bad sampling based estimates of the gradient and the offsets. We therefore trained centered RBMs with 4 visible and 4 hidden units on the 2x2 *Bars & Stripes* data set using either the exact gradient where only the offsets $\langle \mathbf{v} \rangle_m$ and $\langle \mathbf{h} \rangle_m$ were approximated by samples or using PT estimates of the gradient while $\langle \mathbf{v} \rangle_m$ and $\langle \mathbf{h} \rangle_m$ were calculated exactly.

The results are shown in Figure 8.2. If the true model expectations are used for the offsets instead of the sample approximations no divergence for aa is observed when used with PT. Interestingly, the divergence is also prevented if one calculates the exact gradient while still approximating the offsets by samples. Thus, the divergence behavior is induced by the combination of the bad approximations of the offsets and the gradient.

Additionally, the left plot in Figure 8.2 illustrates that centered RBMs outperform normal binary RBMs also if the exact gradient is used. This emphasizes that the worse performance of normal binary RBMs is caused by the properties of its gradient rather than by the gradient approximation.

The results in Table 8.2 demonstrate the flip invariance of the centered RBMs on

ALGORITHM-η	00 *init zero*	00 *init* σ^{-1}
CD-1-0.2	-27.98 ±0.26 (-28.2)	**-21.49** ±1.34 (-22.5)
CD-1-0.1	-28.28 ±0.00 (-28.4)	**-21.09** ±0.97 (-21.6)
CD-1-0.05	-28.28 ±0.00 (-28.3)	**-24.87** ±0.47 (-24.9)
PCD-1-0.2	-28.01 ±0.26 (-28.3)	**-22.45** ±1.00 (-42.3)
PCD-1-0.1	-28.28 ±0.00 (-28.4)	**-21.76** ±0.74 (-26.7)
PCD-1-0.05	-28.28 ±0.00 (-28.3)	**-24.83** ±0.55 (-25.0)
PT_{10}-0.2	-28.01 ±0.27 (-28.2)	**-21.72** ±1.24 (-23.5)
PT_{10}-0.1	-28.28 ±0.00 (-28.4)	**-21.14** ±0.85 (-21.8)
PT_{10}-0.05	-28.28 ±0.00 (-28.3)	**-24.80** ±0.52 (-24.9)

Table 8.3: Average maximum LL for *00* on the *Flipped Shifting Bar* data set, where the visible bias is initialized to zero or to the inverse sigmoid of the data mean.

the *Shifting Bar* data set empirically. While *00* fails to model the flipped version of the data set correctly dd_s^a, aa^b, *d0* have approximately the same performance on the flipped and unflipped data set.

8.6.2 Initialization

The following set of experiments was done to analyze the effects of the different initializations of the parameters discussed in Section 8.4. First, we trained normal binary RBMs (that is *00*) where the visible bias was initialized to zero or to the inverse sigmoid of the data mean. In both cases the hidden bias was initialized to zero. Table 8.3 shows the results for normal binary RBMs trained on the *Flipped Shifting Bar* data set, where RBMs with zero initialization failed to learn the distribution accurately. The RBMs using the inverse sigmoid initialization achieved good performance and therefore seem to be less sensitive to the "difficult" representation of the data. However, the results are not as good as the results of the centered RBMs shown in Table 8.2. The same observations can be made when training RBMs on the *MNIST* data set (see Table 8.4). The RBMs with inverse sigmoid initialization achieved significantly better results than RBMs initialized to zero in the case of PCD and PT, but they are still worse compared to centered RBMs. Furthermore, using the inverse sigmoid initialization allows to achieve similar performance on the flipped and normal version of the *MNIST* data set, while the RBM with zero initialization failed to learn *1-MNIST* at all.

Second, we trained models using the centering versions *dd*, *aa*, and *d0* comparing the initialization suggested in Section 8.4 against the initialization to zero, where we observed that the different ways to initialize had little effect on the performance. In most cases the results show no significant difference in terms of the maximum LL

Algorithm-η	00 *init zero*	00 *init* σ^{-1}
CD-1-0.1	**-165.91** ±1.90 (-168.4)	-167.61 ±1.44 (-168.9)
CD-1-0.05	**-167.68** ±1.66 (-169.0)	-168.72 ±1.36 (-170.8)
CD-1-0.01	-171.29 ±1.49 (-172.4)	**-168.29** ±1.54 (-171.1)
PCD-1-0.1	-160.74 ±4.87 (-169.4)	**-147.56** ±1.17 (-156.3)
PCD-1-0.05	-173.42 ±4.42 (-178.1)	**-144.20** ±0.97 (-149.7)
PCD-1-0.01	-198.00 ±4.78 (-198.4)	**-144.06** ±0.47 (-145.0)
PT_{10}-0.01	-148.76 ±1.15 (-153.6)	**-145.63** ±0.66 (-149.4)

Table 8.4: Average maximum LL *00* on the *MNIST* data set, where the visible bias is initialized to zero or to the inverse sigmoid of the data mean.

Algorithm-η	dd_s^a *init zero*	dd_s^a *init* σ^{-1}
CD-1-0.2	**-20.34** ±0.74 (-20.6)	**-20.42** ±0.80 (-20.8)
CD-1-0.1	**-20.75** ±0.79 (-20.9)	**-20.85** ±0.82 (-21.0)
CD-1-0.05	-23.00 ±0.72 (-23.0)	**-22.63** ±0.66 (-22.6)
PCD-1-0.2	**-21.03** ±0.51 (-30.6)	**-20.97** ±0.65 (-32.3)
PCD-1-0.1	**-20.86** ±0.75 (-23.0)	**-20.72** ±0.50 (-23.1)
PCD-1-0.05	-22.75 ±0.66 (-22.8)	**-22.30** ±0.64 (-22.4)
PT_{10}-0.2	**-20.08** ±0.38 (-20.5)	**-20.25** ±0.55 (-20.7)
PT_{10}-0.1	**-20.56** ±0.69 (-20.7)	**-20.68** ±0.69 (-20.8)
PT_{10}-0.05	-22.93 ±0.72 (-22.9)	**-22.39** ±0.65 (-22.4)

Table 8.5: Average maximum LL for dd_s^a on the *Flipped Shifting Bar* data set, where the visible bias is initialized to zero or to the inverse sigmoid of the data mean.

reached during training with different initializations or slightly better results were found when using the inverse sigmoid, which can be explained by the better starting point yielded by this initialization. See Table 8.5 for the results for dd_s^a on the *Bars & Stripes* data set as an example. We used the inverse sigmoid initialization in the following experiments.

8.6.3 Reparameterization

To investigate the effects of performing the reparameterization before or after the gradient update in the training of centered RBMs (that is, the difference of the algorithm suggested here and the algorithm suggested by Montavon and Müller (2012)), we analyzed the learning behavior of dd_s^b and dd_s^a on all data sets. The results for RBMs trained on the *Bars & Stripes* data set are given in Table 8.6 (top). No significant

Algorithm-η	dd_s^b	dd_s^a
Bars & Stripes		
CD-1-0.1	**-60.34** ±2.18	**-60.41** ±2.08
CD-1-0.05	**-60.19** ±1.98	**-60.25** ±2.13
CD-1-0.01	**-61.23** ±1.49	**-61.22** ±1.49
PCD-1-0.1	**-54.86** ±1.52	**-54.75** ±1.46
PCD-1-0.05	**-53.71** ±1.45	**-53.60** ±1.48
PCD-1-0.01	**-56.68** ±0.74	**-56.68** ±0.73
PT_{10}-0.1	**-51.25** ±1.09	**-51.13** ±0.85
PT_{10}-0.05	**-52.06** ±1.38	**-51.87** ±1.05
PT_{10}-0.01	**-56.72** ±0.77	**-56.73** ±0.77
MNIST		
CD-1-0.1	**-150.60** ±1.55	-150.87 ±1.53
CD-1-0.05	**-150.98** ±1.90	-151.21 ±1.89
CD-1-0.01	**-152.23** ±1.75	-152.39 ±1.81
PCD-1-0.1	**-141.11** ±0.53	**-140.89** ±0.61
PCD-1-0.05	**-139.95** ±0.47	**-140.02** ±0.45
PCD-1-0.01	**-140.67** ±0.46	**-140.68** ±0.42
PT_{10}-0.01	**-141.56** ±0.52	**-141.46** ±0.54

Table 8.6: Average maximum LL on (top) the *Bars & Stripes* data set and (bottom) the *MNIST* data set, using the reparameterization before (dd_s^b) and after (dd_s^a) the gradient update.

difference between the performance of the two centering versions can be observed. The same result was obtained for the *Shifting Bar* and *Flipped Shifting Bar* data set. The results for the *MNIST* data set are shown in Table 8.6 (bottom). Here, no difference could be observed for PCD and PT, but dd_s^b performs slightly better than dd_s^a in the case of CD. Therefore, we reparameterize the RBMs before the gradient update in the remainder of this work.

8.6.4 Usage of an exponentially moving average

We have analyzed the impact of using an exponentially moving average with a sliding factor of 0.01 for the estimation of the offset parameters. Figure 8.3 (left) illustrates on the *Bars & Stripes* data set that the learning curves of the different models become almost equivalent when using an exponentially moving average. The maximum LL values reached are the same whether an exponentially moving average is used or not,

which can also be seen by comparing the results in Table 8.1 and Table 8.2 with those in Table 8.7.

Interestingly, when training an RBM using PT based on the enhanced gradient, an exponentially moving average prevents the observed divergence of the LL. As an example see the learning curves for the *Bars & Stripes* data set in Figure 8.3 (left) in comparison to learning curves for training without exponentially moving average in Figure 8.1 (right). The results can be explained by the more robust approximations of the offsets induced by the smoothing effect of the exponentially moving average. This is coherent with the findings described in Section 8.6.2 where we have observed that the divergence is prevented when using the true expectations.

In the previous experiments *dd* was used with an exponentially moving average as suggested for this centering variant by Montavon and Müller (2012). Note however, that in batch learning when $\langle \mathbf{v} \rangle_d$ is used as visible offset this value stays constant such that an exponentially moving average has no effect. More generally, if the training data and thus $\langle \mathbf{v} \rangle_d$ is known in advance the visible offset should be fixed to this value independent of whether batch, mini-batch or online learning is used. However, the use of an exponentially moving average for approximating $\langle \mathbf{v} \rangle_d$ is reasonable if the training data is not known in advance, as well as for the approximation of the mean $\langle \mathbf{h} \rangle_d$ of the hidden representation.
In our experiments, *dd* does not suffer from the divergence problem when PT is used for sampling, even without exponentially moving average, as can be seen in Figure 8.3 (right) for example. We did not even observe the divergence without a moving average in the case of mini-batch learning. Thus, *dd* seems to be generally more stable than the other centering variants.

8.6.5 Other choices for the offsets

As discussed in Section 8.3, any offset value between 0 and 1 guarantees the flip invariance property as long as it flips simultaneously with the data. An intuitive and constant choice is to set the offsets to 0.5, which has also been proposed by Ollivier et al. (2013) to yield a symmetric variant of the energy of RBMs. This leads to comparable LL values on flipped and unflipped data sets. However, if the data set is unbalanced in the amount of zeros and ones like *MNIST*, the performances is always worse compared to that of a normal RBM on the version of the data set having less ones than zeros. Therefore, fixing the offset values to 0.5 cannot be considered as an alternative for centering using expectation values over the data or model distribution.

In Section 8.3 we mentioned the existence of alternative offset parameters which lead to the same updates for the weights as the enhanced gradient. Setting $\boldsymbol{\mu} = \langle \mathbf{v} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_m$ seems reasonable since the data mean is usually known in advance.

Algorithm-η	aa_s^b	dd_s^b	dm_s^b
Bars & Stripes			
CD-1-0.1	**-60.09** ±2.02 (-69.6)	**-60.34** ±2.18 (-69.9)	**-60.35** ±1.99 (-68.8)
CD-1-0.05	**-60.31** ±2.10 (-64.2)	**-60.19** ±1.98 (-63.6)	**-60.25** ±2.13 (-64.2)
CD-1-0.01	**-61.22** ±1.50 (-61.3)	**-61.23** ±1.49 (-61.3)	**-61.23** ±1.49 (-61.3)
PCD-1-0.1	**-54.78** ±1.63 (-211.7)	**-54.86** ±1.52 (-101.0)	**-54.92** ±1.49 (-177.3)
PCD-1-0.05	**-53.81** ±1.58 (-89.9)	**-53.71** ±1.45 (-67.7)	**-53.88** ±1.54 (-83.3)
PCD-1-0.01	**-56.48** ±0.74 (-56.7)	-56.68 ±0.74 (-56.9)	**-56.47** ±0.74 (-56.6)
PT_{10}-0.1	**-51.20** ±1.11 (-52.4)	-51.25 ±1.09 (-52.3)	**-51.10** ±1.02 (-52.5)
PT_{10}-0.05	**-51.99** ±1.39 (-52.6)	-52.06 ±1.38 (-52.6)	**-51.82** ±1.05 (-52.4)
PT_{10}-0.01	**-56.65** ±0.77 (-56.7)	-56.72 ±0.77 (-56.7)	**-56.67** ±0.77 (-56.7)
Flipped Shifting Bar			
CD-1-0.2	**-20.36** ±0.74 (-20.7)	**-20.32** ±0.69 (-20.6)	**-20.32** ±0.70 (-20.6)
CD-1-0.1	**-20.80** ±0.76 (-20.9)	**-20.86** ±0.81 (-21.0)	**-20.69** ±0.76 (-20.8)
CD-1-0.05	**-22.58** ±0.64 (-22.6)	-22.64 ±0.69 (-22.7)	-22.94 ±0.73 (-23.0)
PCD-1-0.2	**-21.00** ±0.65 (-41.5)	**-20.96** ±0.49 (-31.0)	**-21.00** ±0.68 (-38.3)
PCD-1-0.1	**-20.75** ±0.53 (-23.4)	**-20.76** ±0.53 (-22.8)	**-20.88** ±0.70 (-23.2)
PCD-1-0.05	**-22.28** ±0.68 (-22.3)	**-22.29** ±0.64 (-22.3)	-22.68 ±0.65 (-22.7)
PT_{10}-0.2	**-20.14** ±0.45 (-20.7)	**-20.31** ±0.61 (-20.7)	**-20.07** ±0.38 (-20.5)
PT_{10}-0.1	**-20.42** ±0.51 (-20.7)	**-20.46** ±0.56 (-20.6)	**-20.60** ±0.72 (-20.8)
PT_{10}-0.05	**-22.36** ±0.64 (-22.4)	**-22.39** ±0.69 (-22.4)	-22.86 ±0.70 (-22.9)
MNIST			
CD-1-0.1	**-150.61** ±1.52 (-153.8)	**-150.60** ±1.55 (-153.9)	**-150.50** ±1.48 (-153.6)
CD-1-0.05	**-151.11** ±1.55 (-153.2)	-150.98 ±1.90 (-153.8)	**-150.80** ±1.92 (-153.5)
CD-1-0.01	**-152.83** ±2.42 (-153.3)	**-152.23** ±1.75 (-152.6)	**-152.17** ±1.72 (-152.5)
PCD-1-0.1	**-141.10** ±0.64 (-145.4)	**-141.11** ±0.53 (-145.7)	**-140.99** ±0.56 (-144.8)
PCD-1-0.05	**-140.01** ±0.58 (-142.9)	**-139.95** ±0.47 (-142.6)	**-139.94** ±0.46 (-142.7)
PCD-1-0.01	**-140.85** ±0.47 (-141.6)	**-140.67** ±0.46 (-141.4)	**-140.72** ±0.39 (-141.5)
PT_{10}-0.01	-142.32 ±0.47 (-145.7)	**-141.56** ±0.52 (-143.3)	-142.18 ±0.45 (-146.0)

Table 8.7: Average maximum LL on (top) *Bars & Stripes*, (middel) *Flipped Shifting Bar*, and (bottom) *MNIST* when using an exponentially moving average with an sliding factor of 0.01.

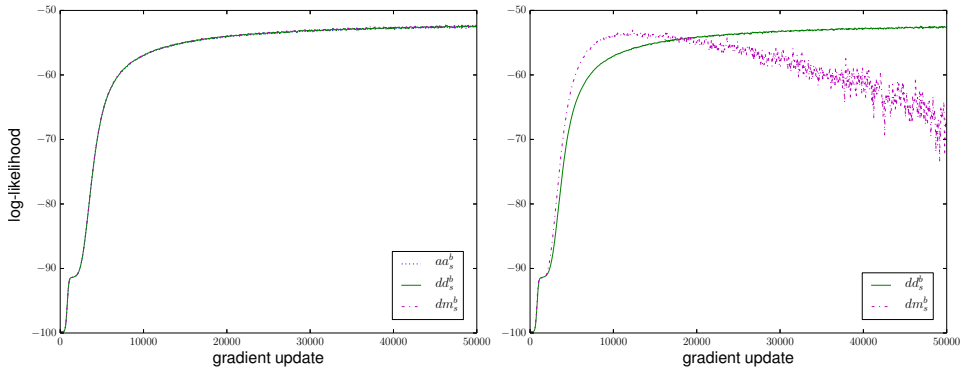


Figure 8.3: Average LL during training on *Bars & Stripes* with the different centering variants, using PT_{10}, and a learning rate of $\eta = 0.05$. (left) When an exponentially moving average with sliding factor of 0.01 was used (where the curves are almost equivalent) and (right) when no exponentially moving average was used.

Following the same notation as above, we refer to centering with this choice of offsets as dm. We trained RBMs with dm_s^b using a sliding factor of 0.01. The results are shown in Table 8.7 and suggest that there is no significant difference between dm_s^b, aa_s^b, and dd_s^b. However, without an exponentially moving average dm^b has the same divergence problems during training with PT_c as aa^b, as shown in Figure 8.3 (right).

We further tried variants like mm, md, $0d$, $m0$ etc. but did not find better performance than that of dd for any of these choices. The variants that subtract an offset from both the visible and the hidden variables always outperformed the variants that subtract an offset only from one type of variables. When the model expectation was used without a exponentially moving average either for $\boldsymbol{\mu}$ or $\boldsymbol{\lambda}$, or for both offsets we always observed the divergence problem.

8.6.6 Experiments with big RBMs

In the previous experiments we trained small models in order to be able to run many experiments and to evaluate the LL exactly. We now want to show that the results observed for the toy problems and for *MNIST* with RBMs with 16 hidden units carry over to more realistic settings. We therefore trained RBMs with 500 hidden units with *00*, *d0*, dd_s^b, or aa_s^b on *MNIST* using the training setup described in Section 8.5.2.

Figure 8.4 shows the average LL over 25 trials for PCD-1 and PT_{20} for the different centering versions, where the LL was estimated every 10th epoch using AIS. Both variants dd_s^b and aa_s^b reach significantly higher LL values than *00* and *d0*. The standard

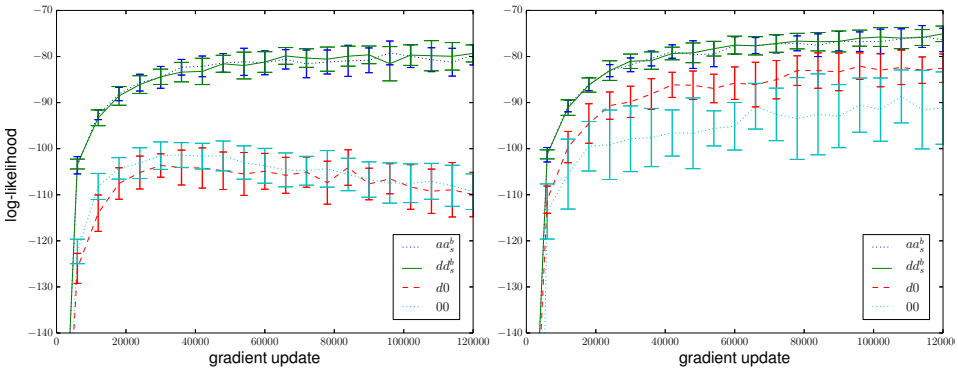


Figure 8.4: Average LL during training on *MNIST* with the different centering variants with 500 hidden units, using a learning rate of $\eta = 0.01$, and a sliding factor of 0.01. (left) When using PCD_1 and (right) when using PT_{20} for sampling. The error bars indicate the standard deviation of the LL over the 25 trials.

deviation over the 25 trials indicated by the error bars is smaller for dd_s^b and aa_s^b than for *00* and *d0*, especially when PT_{20} is used for sampling. Furthermore, *00* and *d0* show divergence already after 30.000 gradient updates when PCD-1 is used, while no divergence can be observed for dd_s^b and aa_s^b after 120.000 gradient updates.

To our knowledge, the best reported performance of an RBM with 500 hidden units trained carefully on *MNIST* was an average LL per sample of -84 (Salakhutdinov, 2008; Salakhutdinov and Murray, 2008; Tang and Sutskever, 2011; Cho et al., 2013b).[2] In our experience, choosing the correct training setup and using additional modifications of the update rule like a momentum term, weight decay, and an annealing learning rate is essential to reach this LL value with normal binary RBMs. However, in order to get an unbiased comparison of the different centering versions, we did not use any additional modification of the update rule. This explains why *00* reaches only a lower LL per sample in our experiments. *d0* however, reaches a value of -84 when PT is used for sampling, and dd_s^b and aa_s^b reach even higher values around -80 with PCD-1 and -75 with PT_{20}. Consistent with the results on small models, the results for bigger RBMS reflect the superiority of dd_s^b and aa_s^b over *d0* and *00*. This supports our statement that centering visible and hidden units in RBMs is important for yielding good models.

One explanation why centering works has been provided by Montavon and Müller

[2]Note, that the binarization of *MNIST* is often done by treating the gray values (normalized to values in $[0, 1]$) as probabilities and sampling the binary values accordingly. Furthermore, RBMs may also be trained on the gray values directly. This makes the likelihood values reported for MNIST experiments difficult to compare across studies.

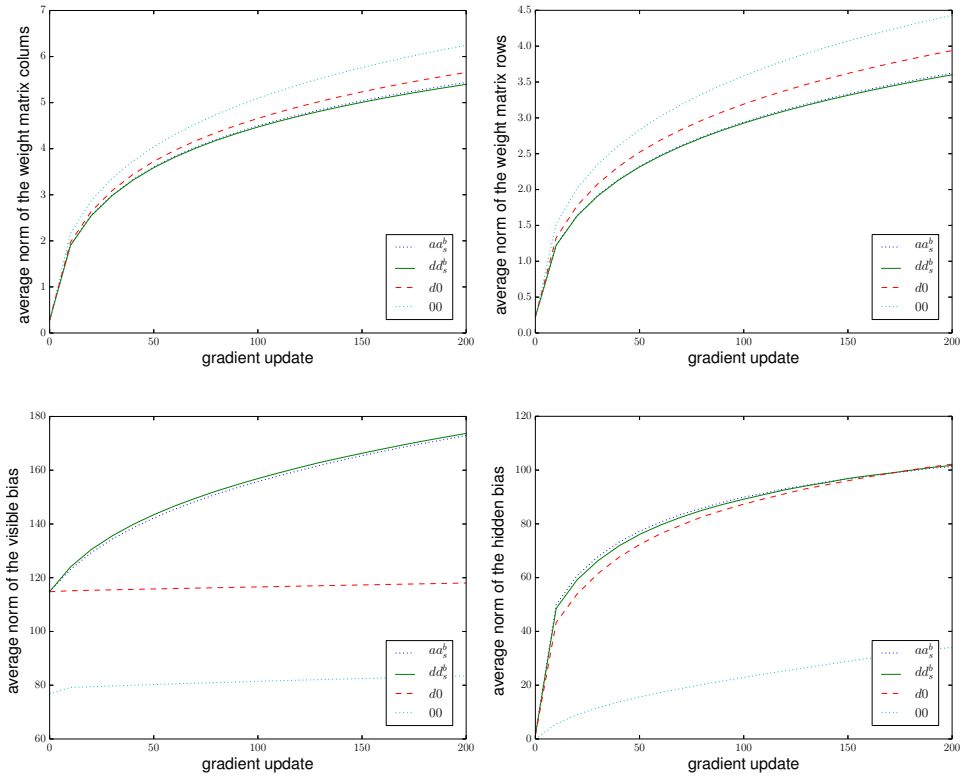


Figure 8.5: Evolution of the average Euclidean norm of the parameters of the RBMs with 500 hidden units trained on *MNIST*, for (top, left) the weight matrix columns, (top, right) the weight matrix rows, (bottom, left) the visible bias, (bottom, right) and for the hidden bias.

(2012), who found that centering leads to an initially better conditioned optimization problem. Furthermore, Cho et al. (2011) has shown that when the enhanced gradient is used the update directions for the weights are less correlated than when the standard gradient is used, which allows one to learn more meaningful features.

From our analysis in Section 8.3 we know that centered RBMs and normal RBMs belong to the same model class and therefore the reason why centered RBMs outperform normal RBMs can indeed only be due to the optimization procedure. Furthermore, one has to keep in mind that in centered RBMs the variables mean values are explicitly stored in the corresponding offset parameters, or if the centered gradient is used for training normal RBMs the mean values are transferred to the corresponding bias parameters. This allows the weights to model second and higher order statistics

right from the start, which is in contrast to normal binary RBMs where weights usually capture parts of the mean values. To support this statement empirically, we calculated the average weight and bias norms during training of the RBMs with 500 hidden units on *MNIST* using the standard and the centered gradient. The results are shown in Figure 8.5, where it can be seen that the row and column norms of the weight matrix for dd_s^b, aa_s^b, and *d0* are consistently smaller than for *00*. At the same time the bias values for dd_s^b, aa_s^b, and *d0* are much bigger than for *00*, indicating that the weight vectors of *00* model information that could potentially be modeled by the bias values. Interestingly, the curves for all parameters of dd_s^b and aa_s^b show the same logarithmic shape, while for *d0*, *00* the visible bias norm does not change significantly. It seems that the bias values did not adapt properly during training. Comparing, *d0* with dd_s^b and aa_s^b, the weight norms are slightly bigger and the visible bias is much smaller for *d0*, indicating that it is not sufficient to center only the visible variables and that visible and hidden bias influence each other. The dependence of the hidden mean and visible bias can also be seen from equation (8.8) where the transformation of the visible bias depends on the offset of the hidden variables.

8.6.7 Comparision to the natural gradient

The results of the previous section indicate that one explanation for the better performance of the centered gradient compared to the standard gradient is the decoupling of the bias and weight parameters. As described in Section 8.2.2 the natural gradient is independent of the parameterization of the distribution. Thus, it is also independent of which parameters store the mean information and should not suffer from the described bias-weight coupling problem. That is why we expect the direction of the centered gradient to be closer to the direction of the natural gradient than the direction of the standard gradient.

To verify this hypothesis empirically, we trained small RBMs with 4 visible and 4 hidden units using the exact natural gradient on the 2x2 *Bars & Stripes* data set. After each gradient update the different exact gradients were calculated and the angle between the centered and the natural gradient as well as the angle between the standard and the natural gradient were evaluated. The results are shown in Figure 8.6 where the top left plot shows the evolution of the average LL when the exact natural gradient is used for training with different learning rates. The right plot shows the average angles between the different gradients during training when the natural gradient is used for training with a learning rate of 0.1. The angle between centered and natural gradient is consistently much smaller than the angle between standard and natural gradient. Comparable result can also be observed for the *Shifting & Bars* data set and when the standard, or centered gradient is used for training.

Notice, how fast the natural gradient reaches a value very close to the theoretical LL upper bound of -13.86 even for a learning rate of 0.1. This verifies empirically the theoretical statement that the natural gradient is clearly the update direction of choice, which should be used if it is tractable. To further emphasize how quick the natural gradient converges, we compared the average LL evolution of the standard, centered and natural gradient, as shown in Figure 8.6 (bottom). Although much slower than the natural gradient, the centered gradient reaches the theoretical upper bound of the LL. The standard gradient seems to saturate on a much smaller value, showing again the inferiority of the standard gradient even if it is calculated exactly and not only approximated.

The results support our assumption that the centered gradient is closer to the natural gradient and is therefore preferable over the standard gradient.

8.7 Conclusion

This work discusses centered binary RBMs, where centering corresponds to subtracting offset parameters from visible and hidden variables. Our theoretical analysis yielded the following results

1. Centered RBMs and normal RBMs are different parameterizations of the same model class.

2. Training a centered RBM can be reformulated to training a normal binary RBM with a new parameter update, which we refer to as centered gradient.

3. From this new formulation follows that the enhanced gradient is just a particular form of centering. That is, the centered gradient becomes equivalent to the enhanced gradient by setting the visible and hidden offsets to the average over model and data mean of the corresponding variable.

4. The LL gradient of centered RBMs is invariant under simultaneous flip of variables and offsets, for any offset value in the range of zero to one. This leads to a desired invariance to changes of the data representation of the generative performance of the model.

Due to the structural similarity these results also extend to DBMs.

Our empirical analysis yielded the following results

5. Centered RBMs reach significantly higher LL values than normal binary RBMs. As an example, centered RBMs with 500 hidden units achieved an average test LL of -76 on *MNIST* compared to a reported value of -84 for carefully trained

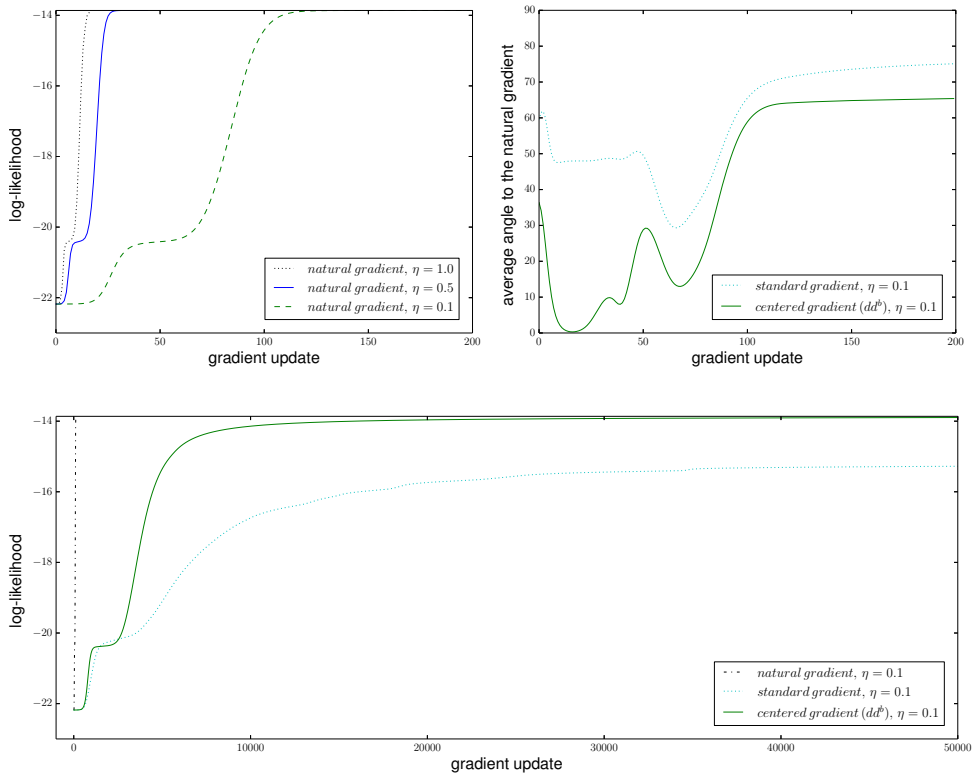


Figure 8.6: Comparison of the centered gradient, standard gradient, and natural gradient for RBMs with 4 visible and 4 hidden units trained on *Bars & Stripes 2x2*. (top, left) The average LL evolution over 25 trials when the natural gradient is used for training with different learning rates, (top, right) the average angle over 25 trials between the natural and standard gradient as well as natural and centered gradient when a learning rate of 0.1 is used, and (bottom) average LL evolution over 25 trials when either the natural gradient, standard gradient or centered gradient is used for training.

normal binary RBMs (Salakhutdinov, 2008; Salakhutdinov and Murray, 2008; Tang and Sutskever, 2011; Cho et al., 2013b).

6. Initializing the bias parameters such that the RBM is initially centered can already improve the performance of a normal binary RBM. However, this initialization leads to a performance still worse compared to the performance of a centered RBM and thus it is not an alternative to centering.

7. Optimal performance of centered RBMs is achieved when both, visible and hidden

variables are centered and the offsets are set to their expectations under the data or model distribution.

8. Using the expectation under the model distribution (as for the enhanced gradient for example) can lead to a severe divergence of the LL when PT_c is used for sampling.

9. This can be prevented when an exponentially moving average for the approximations of the offset values is used.

10. Training centered RBMs leads to smaller weight norms and larger bias norms compared to normal binary RBMs. This supports the hypothesis that when using the standard gradient the mean value is modeled by both weights and biases, while when using the centered gradient the mean values are explicitly modeled by the bias parameters.

11. The direction of the centered gradient is closer to the natural gradient than that of the standard gradient.

All results clearly support the superiority of centered RBMs.

Thus, our work shows that binary RBMs should always be centered and that the expectation under the data distribution is a proper choice for visible and hidden offsets.

8.8 Appendix

Proof of invariance for the centered RBM gradient

In the following we show that the gradient of centered RBMs is invariant to flips of the variables if the corresponding offset parameters flip as well. Since training a centered RBM is equivalent to training a normal binary RBM using the centered gradient (see the proof below), the proof also holds for the centered gradient.

We begin by formalizing the invariance property in the following definitions.

Definition 8.1. *Let there be an RBM with visible variables* $\mathbf{V} = (V_1, \ldots, V_m)$ *and hidden variables* $\mathbf{H} = (H_1, \ldots, H_n)$*. The variables* V_i *and* H_j *are called flipped if they take the values* $\tilde{v}_i = 1 - v_i$ *and* $\tilde{h}_j = 1 - h_j$ *for any given states* v_i *and* h_j*.*

Definition 8.2. *Let there be a binary RBM with parameters* $\boldsymbol{\theta}$ *and energy* E *and another binary RBM with parameters* $\tilde{\boldsymbol{\theta}}$ *and energy* $\tilde{E}$ *where some of the variables are flipped, such that*

$$E(\mathbf{v}, \mathbf{h}) = \tilde{E}(\tilde{\mathbf{v}}, \tilde{\mathbf{h}}) \ , \tag{8.14}$$

for all possible states $(\mathbf{v}, \mathbf{h})$ *and corresponding flipped states* $(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})$*, where* $\tilde{v}_i = 1 - v_i$*,* $\tilde{h}_j = 1 - h_j$*, if* V_i *and* H_j *are flipped, and* $\tilde{v}_i = v_i$*,* $\tilde{h}_j = h_j$*, otherwise. The gradient* $\nabla\boldsymbol{\theta}$ *is called* ***flip-invariant*** *or* ***invariant to the flips of the variables*** *if* (8.14) *still holds after updating* $\boldsymbol{\theta}$ *and* $\tilde{\boldsymbol{\theta}}$ *to* $\boldsymbol{\theta}+\eta\nabla\boldsymbol{\theta}$ *and* $\tilde{\boldsymbol{\theta}}+\eta\nabla\tilde{\boldsymbol{\theta}}$*, respectively, for an arbitrary learning rate* η*.*

We can now state the following theorem.

Theorem 8.1. *The gradient of centered RBMs is invariant to flips of arbitrary variables* $V_{i_1}, \dots V_{i_r}$ *and* $H_{j_1}, \dots H_{j_s}$ *with* $\{i_1, \dots, i_r\} \subset \{1, \dots, m\}$ *and* $\{j_1, \dots, j_s\} \subset \{1, \dots, n\}$ *if the corresponding offset parameters* $\mu_{i_1}, \dots \mu_{i_r}$ *and* $\lambda_{j_1}, \dots \lambda_{j_s}$ *flip as well, that is if* $\tilde{v}_i = 1 - v_i$ *implies* $\tilde{\mu}_i = 1 - \mu_i$ *and* $\tilde{h}_j = 1 - h_j$ *implies* $\tilde{\lambda}_j = 1 - \lambda_j$ *.*

Proof. Let there be a centered RBM with parameters $\boldsymbol{\theta}$ and energy E and another centered RBM where some of the variables are flipped with parameters $\tilde{\boldsymbol{\theta}}$ and energy $\tilde{E}$, such that $E(\mathbf{v}, \mathbf{h}) = \tilde{E}(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})$ for any $(\mathbf{v}, \mathbf{h})$ and corresponding $(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})$. W.l.o.g. it is sufficient to show the invariance of the gradient when flipping only one visible variable V_i, one hidden variable H_j, or both of them, since each derivative with respect to a single parameter can only be affected by the flips of at most one hidden and one visible variable, which follows from the bipartite structure of the model.

We start by investigating how the energy changes when the variables are flipped. For this purpose we rewrite the energy in Equation (8.1) in summation notation given by

$$E(\mathbf{v}, \mathbf{h}) \overset{(8.1)}{=} -\sum_i (v_i - \mu_i)\, b_i - \sum_j (h_j - \lambda_j)\, c_j - \sum_{ij} (v_i - \mu_i)\, w_{ij}\, (h_j - \lambda_j) \quad . \tag{8.15}$$

To indicate a variable flip we introduce the binary parameter f_i that takes the value 1 if the corresponding variable V_i and the corresponding offset μ_i are flipped and 0 otherwise. Similarly, $f_j = 1$ if H_j and λ_j are flipped and $f_j = 0$ otherwise. Now we use $\mathcal{E}^{f_i=1 \wedge f_j=1}$ to denote the terms of the energy (8.15) that are affected by a flip of the variables V_i and H_j. Analogously, $\mathcal{E}^{f_i=1 \wedge f_j=0}$ and $\mathcal{E}^{f_i=0 \wedge f_j=1}$ denote the terms affected by a flip of either V_i or H_j respectively. For flipped values $\tilde{v}_i$, $\tilde{h}_j$ these terms

get

$$\begin{aligned}
\mathcal{E}^{f_i=1\wedge f_j=1} \overset{(8.15)}{=} & -(\tilde{v}_i-\tilde{\mu}_i)b_i-(\tilde{v}_i-\tilde{\mu}_i)\sum_{k\neq j}w_{ik}(h_k-\lambda_k) \\
& -(\tilde{h}_j-\tilde{\lambda}_j)c_j-(\tilde{h}_j-\tilde{\lambda}_j)\sum_{u\neq i}w_{uj}(v_u-\mu_u) \\
& -(\tilde{v}_i-\tilde{\mu}_i)w_{ij}(\tilde{h}_j-\tilde{\lambda}_j) \\
= & -((1-v_i)-(1-\mu_i))b_i-((1-v_i)-(1-\mu_i))\sum_{k\neq j}w_{ik}(h_k-\lambda_k) \\
& -((1-h_j)-(1-\lambda_j))c_j-((1-h_j)-(1-\lambda_j))\sum_{u\neq i}w_{uj}(v_u-\mu_u) \\
& -((1-v_i)-(1-\mu_i))w_{ij}((1-h_j)-(1-\lambda_j)) \\
= & (v_i-\mu_i)b_i+(v_i-\mu_i)\sum_{k\neq j}w_{ik}(h_k-\lambda_k) \\
& (h_j-\lambda_j)c_j+(h_j-\lambda_j)\sum_{u\neq i}w_{uj}(v_u-\mu_u) \\
& -(v_i-\mu_i)w_{ij}(h_j-\lambda_j)\ ,
\end{aligned}$$

and analogously

$$\begin{aligned}
\mathcal{E}^{f_i=1\wedge f_j=0} \overset{(8.15)}{=} & -(\tilde{v}_i-\tilde{\mu}_i)\,b_i-(\tilde{v}_i-\tilde{\mu}_i)\sum_j w_{ij}\,(h_j-\lambda_j) \\
= & (v_i-\mu_i)\,b_i+(v_i-\mu_i)\sum_j w_{ij}\,(h_j-\lambda_j)\ ,
\end{aligned}$$

and

$$\mathcal{E}^{f_i=0\wedge f_j=1} \overset{(8.15)}{=} (h_j-\lambda_j)\,c_j+(h_j-\lambda_j)\sum_i w_{ij}\,(v_i-\mu_i)\ .$$

From the facts that the terms differ from the corresponding terms in (8.15) only in the sign and that $E(\mathbf{v},\mathbf{h})=\tilde{E}(\tilde{\mathbf{v}},\tilde{\mathbf{h}})$ holds for any $(\mathbf{v},\mathbf{h})$ and corresponding $(\tilde{\mathbf{v}},\tilde{\mathbf{h}})$, it follows that the parameters $\tilde{\boldsymbol{\theta}}$ must be given by

$$\begin{aligned}
\tilde{w}_{ij}^{f_i\wedge f_j} &= (-1)^{f_i+f_j}w_{ij}\ , && (8.16)\\
\tilde{b}_i^{f_i\wedge f_j} &= (-1)^{f_i}b_i\ , && (8.17)\\
\tilde{c}_j^{f_i\wedge f_j} &= (-1)^{f_j}c_j\ , && (8.18)\\
\tilde{\mu}_i^{f_i\wedge f_j} &= \mu_i\ , \\
\tilde{\lambda}_j^{f_i\wedge f_j} &= \lambda_j\ .
\end{aligned}$$

The LL gradient for the model without flips is given by Equations (8.4) - (8.6). We now consider the LL gradients for the three possible flipped versions. If V_i and H_j are

flipped the derivatives w.r.t. $w_{ij}, b_i,$, and c_j are given by

$$\begin{aligned}
\nabla \tilde{w}_{ij}^{f_i=1 \wedge f_j=1} &= \langle (1 - v_i - (1 - \mu_i))(1 - h_j - (1 - \lambda_j)) \rangle_d \\
&\quad - \langle (1 - v_i - (1 - \mu_i))(1 - h_j - (1 - \lambda_j)) \rangle_m \\
&= \langle (-v_i + \mu_i)(-h_j + \lambda_j) \rangle_d - \langle (-v_i + \mu_i)(-h_j + \lambda_j) \rangle_m \\
&= \langle (v_i - \mu_i)(h_j - \lambda_j) \rangle_d - \langle (v_i - \mu_i)(h_j - \lambda_j) \rangle_m \\
&= (-1)^{1+1} \nabla w_{ij} \ , \\
\nabla \tilde{b}_i^{f_i=1 \wedge f_j=1} &= \langle 1 - v_i - (1 - \mu_i) \rangle_d - \langle 1 - v_i - (1 - \mu_i) \rangle_m \\
&= - \langle v_i \rangle_d + \mu_i + \langle v_i \rangle_m - \mu_i \\
&= (-1)^1 \nabla b_i \ , \\
\nabla \tilde{c}_j^{f_i=1 \wedge f_j=1} &= \langle 1 - h_j - (1 - \lambda_j) \rangle_d - \langle 1 - h_j - (1 - \lambda_j) \rangle_m \\
&= - \langle h_j \rangle_d + \lambda_j + \langle h_j \rangle_m - \lambda_j \\
&= (-1)^1 \nabla c_j \ .
\end{aligned}$$

If V_i is flipped they are given by

$$\begin{aligned}
\nabla \tilde{w}_{ij}^{f_i=1 \wedge f_j=0} &= \langle (1 - v_i - (1 - \mu_i))(h_j - \lambda_j) \rangle_d \\
&\quad - \langle (1 - v_i - (1 - \mu_i))(h_j - \lambda_j) \rangle_m \\
&= \langle (-v_i + \mu_i)(h_j - \lambda_j) \rangle_d - \langle (-v_i + \mu_i)(h_j - \lambda_j) \rangle_m \\
&= - \left(\langle (v_i - \mu_i)(h_j - \lambda_j) \rangle_d - \langle (v_i - \mu_i)(h_j - \lambda_j) \rangle_m \right) \\
&= (-1)^{1+0} \nabla w_{ij} \ , \\
\nabla \tilde{b}_i^{f_i=1 \wedge f_j=0} &= \nabla \tilde{b}_i^{f_i=1 \wedge f_j=1} \\
&= (-1)^1 \nabla b_i \ , \\
\nabla \tilde{c}_j^{f_i=1 \wedge f_j=0} &= \nabla \tilde{c}_i^{f_i=0 \wedge f_j=0} \\
&= (-1)^0 \nabla c_j \ ,
\end{aligned}$$

and due to the symmetry of the model the derivatives if H_j is flipped are given by

$$\begin{aligned}
\nabla \tilde{w}_{ij}^{f_i=0 \wedge f_j=1} &= (-1)^{0+1} \nabla w_{ij} \ , \\
\nabla \tilde{b}_i^{f_i=0 \wedge f_j=1} &= (-1)^0 \nabla b_i \ , \\
\nabla \tilde{c}_j^{f_i=0 \wedge f_j=1} &= (-1)^1 \nabla c_j \ .
\end{aligned}$$

Comparing the results with Equations (8.16) - (8.18) shows that the gradient underlies the same sign changes under variable flips as the parameters. Thus, it holds for the updated parameters that

$$\begin{aligned}
\tilde{w}_{ij}^{f_i \wedge f_j} + \eta \nabla \tilde{w}_{ij}^{f_i \wedge f_j} &\overset{(8.16)}{=} (-1)^{f_i + f_j} \left(w_{ij} + \eta \nabla w_{ij} \right) \ , \\
\tilde{b}_i^{f_i \wedge f_j} + \eta \nabla \tilde{b}_i^{f_i \wedge f_j} &\overset{(8.17)}{=} (-1)^{f_i + f_j} \left(b_i + \eta \nabla b_i \right) \ , \\
\tilde{c}_j^{f_i \wedge f_j} + \eta \nabla \tilde{c}_j^{f_i \wedge f_j} &\overset{(8.18)}{=} (-1)^{f_i + f_j} \left(c_j + \eta \nabla c_j \right) \ ,
\end{aligned}$$

showing that $E(\mathbf{v}, \mathbf{h}) = \tilde{E}(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})$ is still guaranteed and thus that the gradient of centered RBMs is flip-invariant according to Definition 8.2. □

Theorem 8.1 holds for any value from zero to one for μ_i and λ_j, if it is guaranteed that the offsets flip simultaneously with the corresponding variables. In practice one wants the model to perform equivalently on any flipped version of the data set without knowing which version is presented. This holds if we set the offsets to the expectation value of the corresponding variables under any distribution, since when $\mu_i = \sum_{v_i} p(v_i)\, v_i$, flipping V_i leads to $\tilde{\mu_i} = \sum_{v_i} p(v_i)\,(1 - v_i) = 1 - \sum_{v_i} p(v_i)\, v_i = 1 - \mu_i$ and similary for λ_j, h_j.

Due to the structural similarity this proof also holds for DBMs, where we replace $\mathbf{v}$ by the state $\mathbf{h^l}$ of the variables in the lth hidden layer and $\mathbf{h}$ by the state $\mathbf{h^{l+1}}$ of the variables in the $l+1$th hidden layer to prove the invariance property for the derivatives of the parameters connected to layer l and $l+1$.

Derivation of the centered gradient

In the following we show that the gradient of centered RBMs can be reformulated as an alternative update for the parameters of a normal binary RBM, which we name "centered gradient".

A normal binary RBM with energy $E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T\mathbf{b} - \mathbf{c}^T\mathbf{h} - \mathbf{v}^T\mathbf{W}\mathbf{h}$ can be transformed into a centered RBM with energy $\tilde{E}(\mathbf{v}, \mathbf{h}) = -(\mathbf{v} - \boldsymbol{\mu})^T \tilde{\mathbf{b}} - \tilde{\mathbf{c}}^T (\mathbf{h} - \boldsymbol{\lambda}) - (\mathbf{v} - \boldsymbol{\mu})^T \tilde{\mathbf{W}} (\mathbf{h} - \boldsymbol{\lambda})$ by the following parameter transformation

$$\tilde{\mathbf{W}} \overset{(8.7)}{=} \mathbf{W} \ , \tag{8.19}$$

$$\tilde{\mathbf{b}} \overset{(8.8)}{=} \mathbf{b} + \mathbf{W}\boldsymbol{\lambda} \ , \tag{8.20}$$

$$\tilde{\mathbf{c}} \overset{(8.9)}{=} \mathbf{c} + \mathbf{W}^T\boldsymbol{\mu} \ , \tag{8.21}$$

which guarantees that $E(\mathbf{v}, \mathbf{h}) = \tilde{E}(\mathbf{v}, \mathbf{h}) + const$ for all $(\mathbf{v}, \mathbf{h}) \in \{0, 1\}^{n+m}$ and thus that the modeled distribution stays the same.

Updating the parameters of the centered RBM according to Eq. (8.4) – (8.6) with a learning rate η leads to an updated set of parameters $\tilde{\mathbf{W}}_u, \tilde{\mathbf{b}}_u, \tilde{\mathbf{c}}_u$ given by

$$\tilde{\mathbf{W}}_u \overset{(8.4)}{=} \tilde{\mathbf{W}} + \eta(\langle(\mathbf{v} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T\rangle_d - \langle(\mathbf{v} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T\rangle_m) \ , \tag{8.22}$$

$$\tilde{\mathbf{b}}_u \overset{(8.5)}{=} \tilde{\mathbf{b}} + \eta(\langle\mathbf{v}\rangle_d - \langle\mathbf{v}\rangle_m) \ , \tag{8.23}$$

$$\tilde{\mathbf{c}}_u \overset{(8.6)}{=} \tilde{\mathbf{c}} + \eta(\langle\mathbf{h}\rangle_d - \langle\mathbf{h}\rangle_m) \ . \tag{8.24}$$

One can now transform the updated centered RBM back to a normal RBM by applying the inverse transformation to the updated parameters, which finally leads to the

centered gradient.

$$
\begin{aligned}
\mathbf{W}_u &\overset{(8.19)}{=} \tilde{\mathbf{W}}_u \\
&\overset{(8.19),(8.22)}{=} \mathbf{W} + \eta \underbrace{\left(\langle (\mathbf{v}-\boldsymbol{\mu})(\mathbf{h}-\boldsymbol{\lambda})^T \rangle_d) - \langle (\mathbf{v}-\boldsymbol{\mu})(\mathbf{h}-\boldsymbol{\lambda})^T \rangle_m\right)}_{\overset{(8.10)}{=} \nabla_c \mathbf{W}} , \quad (8.25)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{b}_u &\overset{(8.20)}{=} \tilde{\mathbf{b}}_u - \mathbf{W}_u \boldsymbol{\lambda} \\
&\overset{(8.23),(8.25)}{=} \tilde{\mathbf{b}} + \eta(\langle \mathbf{v} \rangle_d - \langle \mathbf{v} \rangle_m) - (\mathbf{W} + \eta \nabla_c \mathbf{W})\boldsymbol{\lambda} \\
&\overset{(8.20)}{=} \mathbf{b} + \mathbf{W}\boldsymbol{\lambda} + \eta(\langle \mathbf{v} \rangle_d - \langle \mathbf{v} \rangle_m) - \mathbf{W}\boldsymbol{\lambda} - \eta \nabla_c \mathbf{W}\boldsymbol{\lambda} \\
&= \mathbf{b} + \eta \underbrace{\left(\langle \mathbf{v} \rangle_d - \langle \mathbf{v} \rangle_m - \nabla_c \mathbf{W}\boldsymbol{\lambda}\right)}_{\overset{(8.11)}{=} \nabla_c \mathbf{b}} , \quad (8.26)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{c}_u &\overset{(8.21)}{=} \tilde{\mathbf{c}}_u - \mathbf{W}_u^T \boldsymbol{\mu} \\
&\overset{(8.24),(8.25)}{=} \tilde{\mathbf{c}} + \eta(\langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m) - (\mathbf{W} + \eta \nabla_c \mathbf{W})\boldsymbol{\mu} \\
&\overset{(8.21)}{=} \mathbf{c} + \mathbf{W}\boldsymbol{\mu} + \eta(\langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m) - \mathbf{W}\boldsymbol{\mu} - \eta \nabla_c \mathbf{W}\boldsymbol{\mu} \\
&= \mathbf{c} + \eta \underbrace{\left(\langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m - \nabla_c \mathbf{W}\boldsymbol{\mu}\right)}_{\overset{(8.12)}{=} \nabla_c \mathbf{c}} . \quad (8.27)
\end{aligned}
$$

The braces in Equation (8.25) - (8.27) mark the centered gradient given by Equations (8.10) - (8.12).

Chapter 9

On Bennett's acceptance ratio for estimating the partition function of RBMs

---

This chapter is based on the manuscript "On bennett's acceptance ratio for estimating the partition function of restricted Boltzmann machines" by O. Krause, A. Fischer and C. Igel, submitted.

Abstract

The normalization constant of a Restricted Boltzmann Machine (RBM) can be estimated using Annealed Importance Sampling (AIS). Given enough computation time and intermediate distributions to sample from, the estimate is reliable and has low variance. Still, AIS requires a large amount of samples and shows large variance when the bridging distributions are too far apart, which makes AIS impractical for, among others, monitoring training progress. We therefore explore alternative techniques from statistical physics for estimating the partition function of RBMs. A unifying framework for deriving these methods including AIS is presented. When applied to RBMs, a technique known as Bennett's Acceptance Ratio method, which has been suggested in the context of RBMs in a previous study, gives superior results and outperforms AIS, especially when only a small number of bridging chains are employed.

9.1 Introduction

Estimating the normalization constant or partition function of an energy-based probabilistic model (i.e., undirected graphical model or Markov random field) is typically a challenging task, because analytical integration is not possible and numerical integration infeasible. This study focuses on Restricted Boltzmann Machines (RBMs, Smolensky, 1986; Hinton, 2002) as a particular class of Markov random fields. The normalization constant is required for computing the (logarithmic) likelihood of the RBM model parameters, which is to be maximized by RBM learning algorithms. This makes it difficult to assess the performance of trained RBMs, to monitor the training process, or to perform likelihood ratio tests.

Annealed Importance Sampling (AIS, Neal, 2001) as well as variants of Bennett's Acceptance Ratio (BAR, Bennett, 1976) also known as bridge sampling (Meng and Wong, 1996) are statistical tools for estimating the fraction of the normalization constants of two distributions p_{ref} and p_{target}. These techniques introduce bridging distributions connecting p_{ref} and p_{target}. If used to estimate the normalization constant of p_{target} (e.g., represented by an RBM), p_{ref} is chosen such that its normalization constant is known. The performance and limitations of AIS for estimating the partition functions of RBMs are well known (Salakhutdinov and Murray, 2008; Schulz et al., 2010). In contrast, to our knowledge there is only one study applying BAR in the context of RBMs: Desjardins et al. (2011) employed BAR in combination with an importance sampling based estimator using samples from previous learning iterations and a Kalman filter like inference procedure. They showed that the resulting – rather complex – estimation procedure can be used to accurately track the partition function during training while producing only little computational overhead. However, the contributions of the individual parts of the proposed algorithm on the performance have remained largely unknown.

In the following, we compare AIS to BAR on a theoretical and empirical level. We introduce a unifying framework from which variants of AIS and BAR can be derived as special instances. Our result is based on a generalization of Crooks' equality obtained in statistical physics (Crooks, 2000). We show that estimators for the fraction of the normalisation constant fall into two categories: the ones requiring unbiased samples from only one distribution, and the others requiring unbiased samples from both distributions, p_{ref} and p_{target}. Using the former category, which includes AIS, is unproblematic because it is usually easy to define p_{ref} to be a known distribution from which unbiased samples can easily be acquired. Getting unbiased samples from both distributions makes it possible to reduce the variance of the estimator. As a special case of this category, BAR was shown to be the maximum likelihood estimator (Shirts et al., 2003). The need for unbiased samples from the target distribution makes using

these estimators challenging, but we will show the benefits in comparison to AIS.

A possible source of unbiased samples from p_{ref} as well as p_{target} is Parallel Tempering (PT, Desjardins et al., 2010b), which is often used for sampling in RBM training. Parallel Tempering introduces parallel Markov chains to foster faster mixing, which leads to increasing sample quality in all chains. One can directly take the samples from the parallel chains used for RBM training as samples from the bridging distributions of the estimator, which allows for efficient normalization constant estimation (see, e.g., Desjardins et al., 2011).

The next section introduces RBMs as well as PT. Crooks' equality is derived in section three and then used to derive generalized versions of AIS and BAR. Section 4 presents an empirical comparison of a set of estimators of the normalization constant, which follow from the theoretical analysis. The results are discussed in section 5 and section 6 gives the conclusions.

9.2 Restricted Boltzmann machines and parallel tempering

An RBM is an undirected graphical model with a bipartite structure (Smolensky, 1986; Hinton, 2002). The standard binary RBM consists of m visible variables $\boldsymbol{V} = (V_1, \dots V_m)$ taking states $\boldsymbol{v} \in \{0,1\}^m$ and n hidden variables $\boldsymbol{H} = (H_1, \dots H_n)$ taking states $\boldsymbol{h} \in \{0,1\}^n$. The joint distribution is a Gibbs distribution $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ with energy $E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h} - \boldsymbol{v}^T \boldsymbol{b} - \boldsymbol{c}^T \boldsymbol{h}$, where $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$ are the weight matrix and the visible and hidden bias vectors, respectively. The normalization constant $Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ (also referred to as partition function) is typically unknown, because it is calculated by summing over all possible states of hidden and visible units, which is exponential in $\min\{n, m\}$.

It is not possible to sample from the Gibbs distribution of an RBM directly, instead Markov chain Monte Carlo methods are applied. The most common sampling technique is block Gibbs sampling, where a Markov chain $(\boldsymbol{X}^{(t)})_{t \geq 0}$ with $\boldsymbol{X}^{(t)} = (\boldsymbol{V}^{(t)}, \boldsymbol{H}^{(t)})$ starting from an arbitrary state $\boldsymbol{x}^{(0)} = (\boldsymbol{v}^{(0)}, \boldsymbol{h}^{(0)})$ is generated using the transition operator $T((\boldsymbol{v}^{(t)}, \boldsymbol{h}^{(t)}) \to (\boldsymbol{v}^{(t+1)}, \boldsymbol{h}^{(t+1)})) = p(\boldsymbol{v}^{(t+1)} | \boldsymbol{h}^{(t+1)}) p(\boldsymbol{h}^{(t+1)} | \boldsymbol{v}^{(t)})$. With $t \to \infty$ the distribution of $\boldsymbol{x}^{(t)}$ approaches $p(\boldsymbol{v}, \boldsymbol{h})$, but getting close to the stationary distribution usually requires a lot of iterations as the samples produced by block Gibbs sampling are often strongly correlated. The speed with which the chain approaches the stationary distribution is called the mixing rate.

The arguably most promising sampling technique used for RBM training so far is PT (Desjardins et al., 2010b). It introduces supplementary Gibbs chains that sample form more and more smoothed replicas of the RBM distribution. Given a ordered

set of inverse temperatures $0 < \beta_0 < \beta_i < \cdots < \beta_N = 1$, PT maintains a set of N Markov chains with stationary distributions $p_i(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z_i} e^{-\beta_i E(\boldsymbol{v},\boldsymbol{h})}$, $i = 0, \ldots, N$, where Z_i denotes the corresponding partition function. In each step, the algorithm runs k (usually $k = 1$) Gibbs sampling steps in each of the N tempered Markov chains yielding samples $(\boldsymbol{v}_1, \boldsymbol{h}_1), ..., (\boldsymbol{v}_N, \boldsymbol{h}_N)$. After this, two neighbouring Gibbs chains with inverse temperatures β_i and β_{i+1} may exchange samples $(\boldsymbol{v}_i, \boldsymbol{h}_i)$ and $(\boldsymbol{v}_{i+1}, \boldsymbol{h}_{i+1})$ with an exchange probability based on the Metropolis ratio

$$\min\left(1, \frac{p_i\big(\boldsymbol{v}_{i+1}, \boldsymbol{h}_{i+1}\big) p_{i+1}\big(\boldsymbol{v}_i, \boldsymbol{h}_i\big)}{p_i\big(\boldsymbol{v}_i, \boldsymbol{h}_i\big) p_{i+1}\big(\boldsymbol{v}_{i+1}, \boldsymbol{h}_{i+1}\big)}\right) . \tag{9.1}$$

After performing this swaps between chains, the (eventually exchanged) sample $(\boldsymbol{v}_1, \boldsymbol{h}_1)$ of the chain with inverse temperature $\beta_N = 1$ is taken as a sample from the model distribution.

9.3 Optimal estimators of the normalisation constant for a given sampler

This section introduces an unifying framework for estimating normalization constants of Markov random fields. Then different estimators are derived in the subsections.

Let $p_{\text{ref}} = p_0, p_1, \ldots, p_N = p_{\text{target}}$ be a set of Gibbs distributions over some state space Ω with $p_i : \Omega \to \mathbb{R}$, $p_i(x) = \frac{1}{Z_i} e^{-E_i(x)} = \frac{1}{Z_i} p_i^*(x)$. Our goal is to estimate Z_N/Z_0.[1] Let us now consider a random variable $\boldsymbol{X} = (X_0, X_N, \boldsymbol{Y})$ taking values $\boldsymbol{x} = (x_0, x_N, \boldsymbol{y})$ in an extended state space $\Omega^* = \Omega^2 \times \Theta$, where $\boldsymbol{Y}$ taking values in the state space Θ is a placeholder for any set of additional variables an actual estimation method may require. Assume that we can use the set of Gibbs distributions $p_0, p_1, \ldots, p_N$ to construct a pair of distributions $\mathcal{P}$ and $\tilde{\mathcal{P}}$ on Ω^* with $\mathcal{P}[\boldsymbol{x}] = \mathcal{P}[\boldsymbol{y}, x_N | x_0] p_0(x_0)$ and $\tilde{\mathcal{P}}[\boldsymbol{x}] = \tilde{\mathcal{P}}[\boldsymbol{y}, x_0 | x_N] p_N(x_N)$. We will call $\mathcal{P}$ the *forward* distribution as it creates samples from p_N given a sample x_0 from p_0 and denote $\tilde{\mathcal{P}}$ accordingly as the *reverse* distribution. From now on, we will use square brackets to distinguish functions involving the extended state space.

It holds

$$\frac{\tilde{\mathcal{P}}[\boldsymbol{x}]}{\mathcal{P}[\boldsymbol{x}]} = \frac{Z_0 \tilde{\mathcal{P}}[\boldsymbol{y}, x_0 | x_N] p_N^*(x_N)}{Z_N \mathcal{P}[\boldsymbol{y}, x_N | x_0] p_0^*(x_0)} = \frac{Z_0}{Z_N} e^{-\mathcal{W}[\boldsymbol{x}]} , \tag{9.2}$$

where we define $\mathcal{W}[\boldsymbol{x}] = -\ln \frac{\mathcal{P}[\boldsymbol{y}, x_N | x_0] p_0^*(x_0)}{\tilde{\mathcal{P}}[\boldsymbol{y}, x_0 | x_N] p_N^*(x_N)}$.

Consider now any function $\mathcal{F}$ on the extended state space Ω^*. We are interested in relating expectations of $\mathcal{F}$ under the forward distribution to expectations of $\mathcal{F}$ under

[1] If we choose p_0 such that Z_0 is easy to compute (e.g., to be uniform), we also get an estimate of Z_N.

the reverse distribution. To ease the notation, we denote expectations by $\langle f \rangle_p = \int p(x) f(x)\, dx$. We can use the basic idea of importance sampling and equation (9.2) to get

$$\tilde{\mathcal{P}}[\boldsymbol{x}]\mathcal{F}[\boldsymbol{x}] = \mathcal{P}[\boldsymbol{x}]\frac{\tilde{\mathcal{P}}[\boldsymbol{x}]}{\mathcal{P}[\boldsymbol{x}]}\mathcal{F}[\boldsymbol{x}] = \frac{Z_0}{Z_N}\mathcal{P}[\boldsymbol{x}]e^{-\mathcal{W}[\boldsymbol{x}]}\mathcal{F}[\boldsymbol{x}] \ .$$

Taking the expectation we arrive at

$$\langle \mathcal{F} \rangle_{\tilde{\mathcal{P}}} = \frac{Z_0}{Z_N} \left\langle \mathcal{F} e^{-\mathcal{W}} \right\rangle_{\mathcal{P}} \ . \tag{9.3}$$

This result generalizes Crooks' equation (Crooks, 2000) to arbitrary sampling distributions.

We are now ready to derive our main result: generalized versions of AIS and Bennett's acceptance ratio method. By setting $\mathcal{F}[\boldsymbol{x}] = 1$ in equation (9.3) and reordering terms we get

$$\frac{Z_N}{Z_0} = \left\langle e^{-\mathcal{W}} \right\rangle_{\mathcal{P}} \ , \tag{9.4}$$

which can be viewed as a generalization of AIS, where the standard formulation of AIS is obtained by a certain choice of $\mathcal{P}$. Instead of fixing $\mathcal{F}$ to a constant, we can try to find the optimal function leading to the asymptotically best estimator. Bennett (1976) approached this problem for a set of only two Gibbs distributions by finding the $\mathcal{F}$ that minimizes the variance of the estimator for sufficiently large sample size. Crooks (2000) transferred the result to more than two Gibbs distributions and Shirts et al. (2003) showed that the same estimator is found using maximum likelihood principles. A generalized version of the proof can be found in the appendix in Section 9.6. It turns out that the maximum likelihood solution C of $\ln(Z_N/Z_0)$ given a set of samples $W_1, \ldots, W_r$ with $W_i = \mathcal{W}[\boldsymbol{x}_i]$ from the forward distribution and a set of samples $\tilde{W}_1, \ldots, \tilde{W}_r$ with $\tilde{W}_j = \mathcal{W}[\tilde{\boldsymbol{x}}_j]$ from the reverse distribution can be found by solving the following equality

$$\sum_{j=1}^{r} \sigma(\tilde{W}_j + C) - \sum_{i=1}^{r} \sigma(-W_i - C) = 0 \ , \tag{9.5}$$

where σ is the logistic function.

Solving (9.5) for different choices of the forward and the reverse distribution can be seen as a general way to yield BAR-like estimators. As we will see later, a certain choice of $\mathcal{P}$ and $\tilde{\mathcal{P}}$ leads to the estimator introduced by Bennett (1976) and generalized by Crooks (2000) and Shirts et al. (2003).

9.3.1 Methods sampling paths

We consider now *forward* paths $\boldsymbol{x} = (x_0, \ldots, x_N)$ on the extended state space $\Omega^* = \Omega^{N+1}$. Every path is created by a Markov chain which starts by sampling the initial

state x_0 from p_0 (e.g., the uniform distribution) and proceeds by sampling states x_i using transition operators $T_i(x_{i-1} \to x_i),\ i = 1, \ldots, N$. We then regard the probability of the path as the probability of $\boldsymbol{x}$ under the forward distribution

$$\mathcal{P}[\boldsymbol{x}] = p_0(x_0) \prod_{i=1}^{N} T_i(x_{i-1} \to x_i) \ . \tag{9.6}$$

We can now change the point of view on the dynamics and consider the *reverse* path formed by sampling x_N from p_N and sampling the distributions backwards using the reversed operators $\tilde{T}_i(x_i \to x_{i-1})$. This leads to the reverse distribution $\tilde{\mathcal{P}}[\boldsymbol{x}] = p_N(x_N) \prod_{i=1}^{N} \tilde{T}_i(x_i \to x_{i-1})$.

Let us now assume that for the sampling operators T_i and $\tilde{T}_i$ holds

$$p_i(x_{i-1}) T_i(x_{i-1} \to x_i) = p_i(x_i) \tilde{T}_i(x_i \to x_{i-1}) \ .$$

This is the exact same requirement as induced by tempered transitions (Salakhutdinov, 2009) and AIS (Neal, 2001). Especially it holds with $T_i = \tilde{T}_i$ if T_i fulfills detailed balance. It also holds if $T_i = Q_1 \ldots Q_k$, each transition operator $Q_j, j = 1, \ldots, k$, fulfills detailed balance, and $\tilde{T}_i = Q_k \ldots Q_1$. This includes block Gibbs sampling as used for RBMs (Salakhutdinov, 2009).

Given this construction, $\mathcal{W}[\boldsymbol{x}]$ simplifies to

$$\mathcal{W}[\boldsymbol{x}] = -\ln \frac{p_N(x_N) \prod_{i=1}^{N} \frac{p_i(x_{i-1})}{p_i(x_i)} T_i(x_{i-1} \to x_i)}{p_0(x_0) \prod_{i=1}^{N} T_i(x_{i-1} \to x_i)} = \sum_{i=0}^{N-1} \left[E_{i+1}(x_i) - E_i(x_i) \right] \ . \tag{9.7}$$

Inserting these and the definition of $\mathcal{P}$ given in (9.6) into (9.4), we arrive at AIS as proposed by Neal (2001). If we instead insert them into equation (9.5) we arrive at Bennett's acceptance ratio method applied to whole paths as proposed by Crooks (2000).

9.3.2 Methods sampling independent paths

So far we assumed that our transition operator generates dependent samples. If we drop this assumption, we can derive another family of estimators. Making samples independent amounts to setting $T_i = \tilde{T}_i = p_i$ and thus we get $P[\boldsymbol{x}] = \prod_{i=0}^{N} p_i(x_i)$.

$\mathcal{W}[\boldsymbol{x}]$ is still defined as in equation (9.7) but if we insert this again into (9.4) we can now factor out the independent terms to arrive at

$$\frac{Z_N}{Z_0} = \int \prod_{i=0}^{N} p_i(x_i) e^{-\sum_{j=0}^{N-1} [E_{j+1}(x_j) - E_j(x_j)]} \, d\boldsymbol{x} = \prod_{i=0}^{N-1} \left\langle e^{E_i - E_{i+1}} \right\rangle_{p_i} \ . \tag{9.8}$$

This result can also be written as the telescope product $\prod_i^{N-1} Z_{i+1}/Z_i$, where every term Z_{i+1}/Z_i is represented by the well known entity $\langle p^*_{i+1}/p^*_i \rangle_{p_i}$. If we additionally

assume that $\mathcal{F}[\boldsymbol{x}] = \prod_{i=0}^{N-1} f_i(x_i)$ and set $f_i(x) = \alpha_i(x) e^{-E_i(x)} = \alpha_i(x) p_i^*(x)$ we arrive at bridge sampling

$$\frac{Z_N}{Z_0} = \prod_{i=0}^{N-1} \frac{\left\langle \alpha_i p_{i+1}^* \right\rangle_{p_i}}{\left\langle \alpha_i p_i^* \right\rangle_{p_{i+1}}} \ .$$

A prominent choice is $\alpha_i^{-1}(x) = \sqrt{p_i^*(x) p_{i+1}^*(x)}$, the geometric mean of both distributions, see the review by Gelman and Meng (1998). Bennett (1976) used $\alpha_i(x) = \frac{1}{Z_{i+1} p_i(x) + Z_i p_{i+1}(x)}$ in the derivation of his estimator minimizing the variance. The BAR estimator he derived generalized to N Gibbs distributions (using independent instead of dependent samples in contrast to Crooks (2000)) is equivalent to the solution found for (9.5) applied to each pair of distributions p_i and p_{i+1} for $i = 0, \ldots, N-1$ separately, leading to maximum likelihood estimates C_i for Z_{i+1}/Z_i. The estimator of Z_N/Z_0 is finally given by $C = \sum_i^N C_i$.

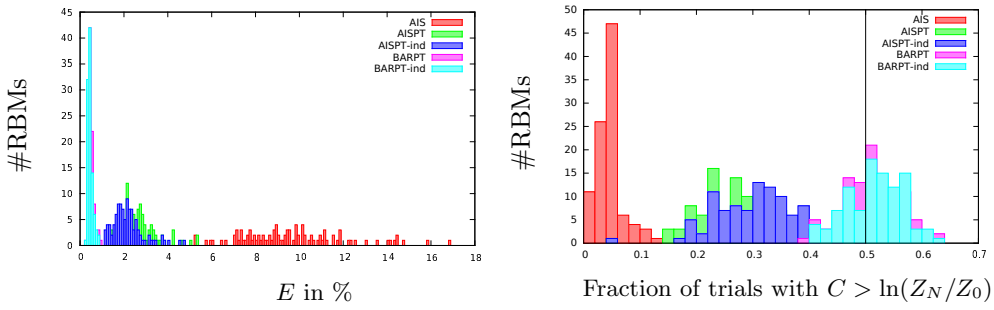
9.4 Experiments and results

We added several partition function estimation methods to an RBM library (the implementation will be made available upon positive evaluation of the manuscript). We compared them on RBMs trained on a number of artificial datasets as well as on the MNIST dataset (LeCun et al., 1998a). In all experiments we considered the task of estimating $\ln(\frac{Z_N}{Z_0})$. We do not report running times, because the runtime is dominated by the sampling procedure and, thus, runtime differences between the different methods are negligible. For the artificial datasets we made sure that our experimental setup always allowed to calculate the exact values of the normalization constants as ground truth. Let C be the estimate of $\ln \frac{Z_N}{Z_0}$. We measured the mean relative error

$$E = \left\langle \left| \frac{C}{\ln \frac{Z_N}{Z_0}} - 1 \right| \right\rangle_{p(C)} = \frac{\left\langle \left| C - \ln(\frac{Z_N}{Z_0}) \right| \right\rangle_{p(C)}}{\left| \ln(\frac{Z_N}{Z_0}) \right|} \ ,$$

which allows us to compare the results of RBMs with different normalization constants. This error measure has the disadvantage that it becomes overly sensitive towards small perturbations when $\ln \frac{Z_N}{Z_0} \to 0$. This is not a problem, because it usually only happens in the first few iterations.

All our experiments were based on the same setup. For a given RBM we took two sets of samples, one created by PT and one using path sampling as in equation (9.6). In both cases, block Gibbs sampling served as transition operator. We chose $\beta_i = 2\sigma(\frac{6i}{N-1}) - 1$, which leads to a majority of chains having inverse temperatures close to 1. We used a short burn-in time for the PT chains of 10% of the number of samples drawn. To make the experiments fair, we increased the number of samples by



(a) Error (b) Error Distribution

Figure 9.1: Histograms of the results of experiment 1 for 100 randomly generated RBMs with 16 visible and 32 hidden units and weights drawn from $\mathcal{N}(0, 0.5)$. The partition function of each RBM was estimated 100 times using 5000 samples for each algorithm. Left: Histogram over the mean errors in %. Right: Histogram over the fraction of trials in which the estimate was bigger than the true value. The black vertical line marks the 50% point.

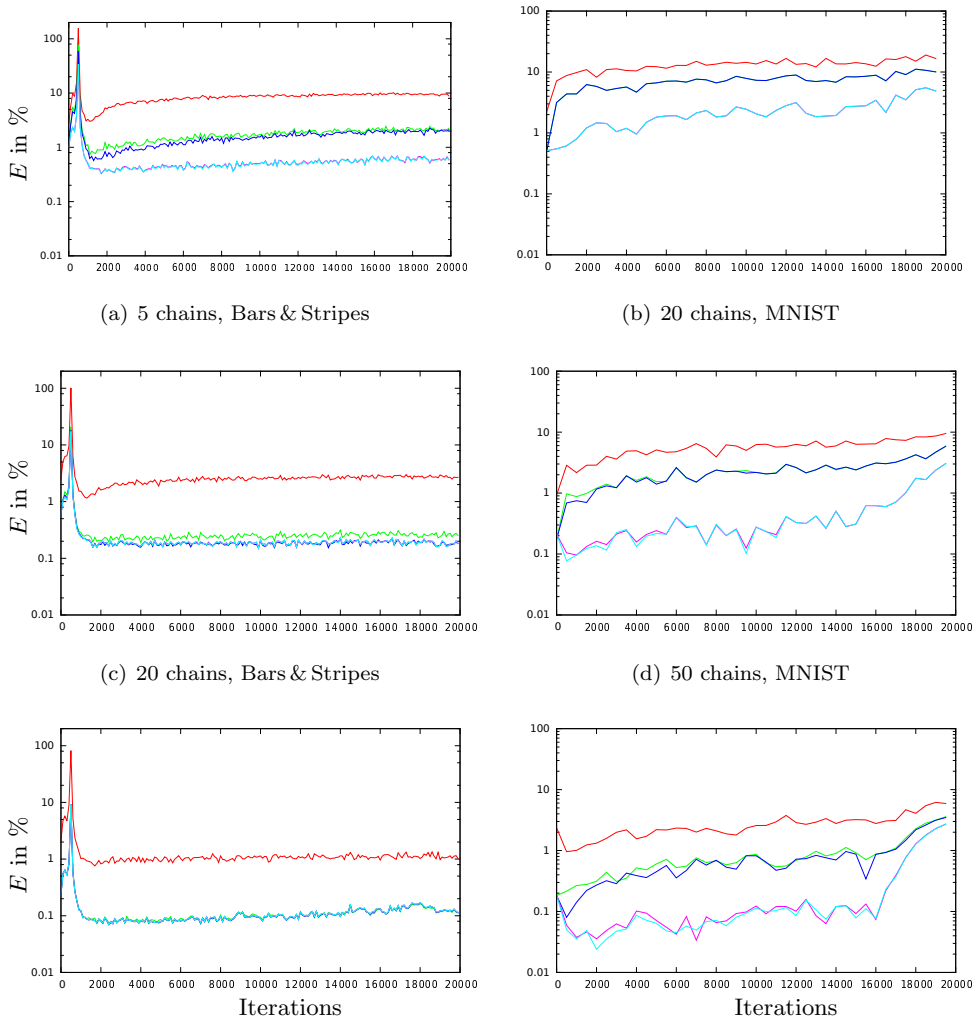
this amount when sampling the paths. Thus, all algorithms used the same number of samples. All algorithms rely on $-\ln p_i^*(\boldsymbol{h})$ instead of $E_i(\boldsymbol{v}, \boldsymbol{h})$ as energy estimate, so we integrate over all possible visible states for every hidden sample.

The dependent path samples were used to calculate the original AIS baseline following equation (9.4). All other algorithms we considered used the same PT samples.

We call AIS using PT samples AISPT, and AIS using the independence assumption of equation (9.8) is called AISPT-ind. Bennett's acceptance ratio using equation (9.5) (i.e., not making use of the independence assumption) is denoted by BARPT and the same estimator using paths of length 1 as described in the end of section 9.3.2 and thus using the independence of the samples is termed BARPT-ind. We are not looking at results of Bennett's acceptance ratio using path sampling, because this would require unbiased samples from p_N, which are not available unless we additionally use PT with many intermediate chains to obtain them.

In **experiment 1**, we generated a set of RBMs with 16 visible and 32 hidden units. The weights of the RBMs were chosen randomly with mean 0 and variance 0.5. This leads to values of $\ln(Z_N/Z_0)$ between 10 and 40. For every RBM we calculated the mean error over a 100 estimates with 5 intermediate distributions and 1000 samples for each of these distributions. Additionally we counted the number of times, the estimate was bigger than the target value to assess the symmetry of the distributions of the estimates. The results for the single RBMs were cumulated in histograms which can be seen in Figure 9.1.

All algorithms using PT samples were better than AIS. Moreover, the BAR variants



(a) 5 chains, Bars & Stripes

(b) 20 chains, MNIST

(c) 20 chains, Bars & Stripes

(d) 50 chains, MNIST

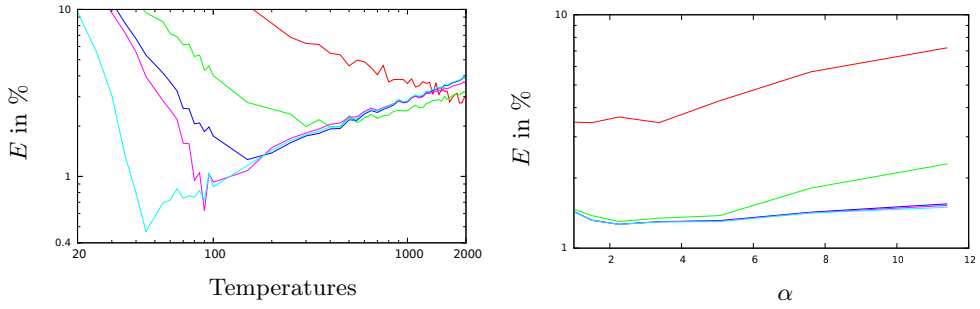
(e) 100 chains, Bars & Stripes

(f) 100 chains, MNIST

Figure 9.2: Mean relative error in percent for the different algorithms during training of RBMs with 16 hidden units. Left: Bars & Stripes, 100 evaluations every 100 iterations. Right: MNIST , 10 evaluations every 500 iterations. Both experiments were carried out with different numbers of chains and 1000 samples per chain and evaluation. We use the same color coding as in Figure 9.1. The curves of BARPT and BARPT-int as well as AISPT and AISPT-int are overlapping in (b).

outperformed the AIS variants. When looking at the error distribution, we see that AIS typically underestimated the normalization constant and only rarely overestimated it. In comparison, BAR showed a symmetric distribution suggesting that the true value is the mode of the distribution.

In **experiment 2**, we analyzed the performance of the estimators over the training



(a) Number of Temperatures (b) Temperature distribution

Figure 9.3: Mean relative error in percent of $\ln(Z_N/Z_0)$ for the CD-25 RBM with 500 hidden units. Every point is the mean of 10 evaluations. We use the same color coding as in Figure 9.1. Left: The number of chains was varied $(20, 25, \ldots, 100, 200, \ldots, 2000)$ and the total amount of samples per run was fixed to 600000. Right: For 500 temperatures, where the distribution of the temperatures was changed.

process when the number of intermediate (i.e, bridging) chains is varied. We trained an RBM with 16 hidden units on the Bars & Stripes (MacKay, 2002) and MNIST dataset. We trained them with 1-step Contrastive Divergence (CD-1, Hinton, 2002) on the full datasets with learning rate 0.1 for 20000 iterations. On Bars & Stripes $\ln(Z_N/Z_0)$ was estimated every 100 iterations 100 times, each time using 1000 samples from each distribution. On MNIST an evaluation was done every 500 iterations with 10 estimates using again 1000 samples per distribution. For Bars & Stripes 5, 20 and 100 chains were considered, while MNIST used 20, 50 and 100. Note that in the case of 100 chains the estimators were based on 100000 samples, which is bigger than the cardinality 2^{16} of the state space of the hidden units. The results, depicted in Figure 9.2, confirmed the findings from the first experiment. In addition, it can be observed that with decreasing number of chains the performance gap between AISPT and BARPT grows.

Experiment 3 investigated the effect of changing the number of temperatures on the quality of the estimates for a bigger RBM while keeping the overall number of samples constant. We used the CD-25 RBM trained by Salakhutdinov and Murray (2008) to model MNIST with 500 hidden units. We varied the number of temperatures between $\{20, 25, \ldots, 100, 200, \ldots, 2000\}$ and used 600000 samples in total. That is, in the case of 5 chains, every chain had 30000 samples, and in the case of 2000 chains only 300 samples were taken from each chain. As ground truth we used the value $\ln(Z_N/Z_0) = -438.72$ reported by Salakhutdinov and Murray (2008). The results are given in Figure 9.3(a).

When the number of chains was small, AIS started with relatively high error rates. When the number of chains was increased, it performed better, in the end even outper-

Burn-in	AIS %	AISPT %	AISPT-ind %	BARPT%	BARPT-ind%
30	3.06324	3.13405	3.83176	3.58562	3.86240
150	2.68570	2.87891	2.98900	2.99325	2.99545
300	2.75816	2.40381	2.45748	2.45838	2.45836

Table 9.1: Error rates of the estimators with 2000 intermediate chains when the burn-in time is prolonged.

forming the other methods. The overall minimum error was achieved by BARPT-ind and BARPT in settings with a relatively low number of chains. The error increased for all methods except from AIS when more chains were used. We analysed this observation and found out that the reason is the bias of the samples gained by PT when having only a very short burn-in time. As we added a burn-in of 10% of the number of samples drawn, we have a burn-in phase of 3000 sampling steps when using 20 intermediate chains, but only of 30 in the case of 2000 chains. We conducted an experiment investigating the error when increasing the burn-in time and found that this reduced the error of all estimators considerably and made the difference between estimators small. The results can be found in Table 9.1.

In **experiment 4** we investigated the effect of the chosen distribution of the temperatures on the quality of the estimates when using 500 chains. We defined a new set of beta values, $\beta_{i,\alpha} = \frac{\sigma(\alpha i/N)-\sigma(0)}{\sigma(\alpha)-\sigma(0)}$, where $\alpha > 0$, $i = 0, \ldots, N$. With small values of α the distribution of temperatures approaches the uniform distribution. Bigger values of α lead to more and more β values being close to 1. We used the same experimental setup as in experiment 3 with a fixed number of 500 temperatures while only changing the distribution of temperatures. To reduce the impact from slow mixing chains, we increased the burn-in time to 100% which amounts to 1200 samples. It turned out that α values close to 7.3 lead to a similar distribution of temperatures as used in experiment 3, see Figure 9.4 for the evaluated temperature distributions.

The results in Figure 9.3(b) show that with growing α the performance of AIS and AISPT decreased considerably, while the results of the other methods remained almost constant. Thus, our initial choice of the temperature distribution was not optimal for AIS. In light of these results, we repeated experiment 1 with a uniform distribution of the temperatures and a set of temperatures resulting from the above formula with $\alpha = 2$. The results (Figure 9.6 in the appendix in Section 9.6) are similar to the one shown in Figure 9.1, albeit less pronounced.

Experiment 5 was designed to compare the performance of the estimators when reusing the samples gained from PT during training for the estimators, instead of starting a new PT chain for every estimate. We trained an RBM with 16 hidden neurons on MNIST with PT using 50 tempered chains. We used the full training set

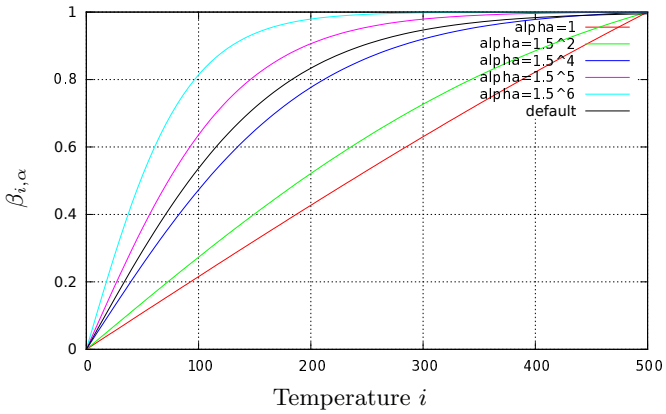


Figure 9.4: Temperature distributions used in experiment 4. The curves for $\alpha = 1.5$ and $\alpha = 1.5^3$ are not shown to avoid a cluttered plot. We refer to the distribution used in all remaining experiments as 'default'.

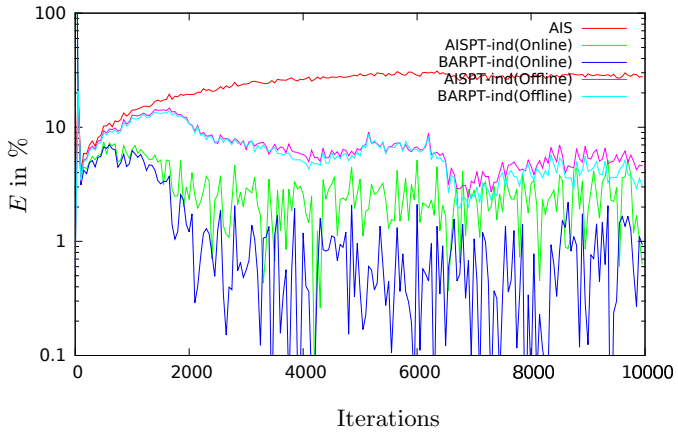


Figure 9.5: Comparison of the errors of the estimators using samples from the PT chain used for training (Online) or using samples gained from a separate PT chain (Offline). Estimators were based on 250000 samples from 50 chains. The mean error of 10 offline trials is compared with one online trial.

but only 5000 PT-samples for the gradient approximation in every learning iteration. We refer to the estimators that reuse the samples from training as online and to the estimators using separate PT chains as offline. As in all other experiments we used 10% burn-in time for the offline estimators. The results can be seen in Figure 9.5

The online estimators got considerably lower error rates compared to their offline counterparts. This is because the persistent PT chains used during training have enough time to mix and thus are much closer to the true distribution.

9.5 Conclusion

This paper introduced a theoretic framework linking the expectation over a distribution of samples generated by a transition operator to the expectation over the distribution induced by the reversed operator. This leads to a generalized form of Crooks' equality, which can be used to devise generalizations of known estimators for the normalization constants of energy-based probabilistic models, including Annealed Importance Sampling (AIS) and Bennett's Acceptance Ratio method (BAR). These generalizations allow the use of different sample sources. We focused on Parallel Tempering (PT) and path sampling for generating samples, but Linked Importance Sampling (Neal, 2005) also fits into this scheme.

In our experiments, we considered estimating the partition functions of RBMs. We compared the AIS- and BAR-based estimators using independent samples from PT with vanilla AIS. All PT based approaches performed better than vanilla AIS. Furthermore, the algorithms based on BAR outperformed all AIS variants.[2] Statistics on the estimation errors showed that AIS estimates are not only worse compared to BAR, but the results are heavily skewed: AIS almost always underestimated the true value. In contrast, the error distribution in the BAR experiments was almost symmetric. Moreover, AIS strongly depends on the right choice of bridging distributions, while BAR worked reliably across a range of temperature choices. The drawback of BAR is that it requires samples from PT and, thus, one has to be careful about the bias of samples when using short burn-in times. This, however, was no problem when using the samples from a persistent PT chain during learning. In summary, we suggest to use BAR with PT to estimate the partition function of an RBM instead of AIS.

[2]The superior performance of BAR is in accordance with the observations by Desjardins et al. (2011), who reported good results when estimating the partition function in an iterative learning task by a sampling procedure using, among others, BAR and PT as building blocks.

9.6 Appendix

Derivation of the maximum likelihood estimate of $\ln(Z_N/Z_0)$

In the following we show that the maximum likelihood estimate C of $\ln(Z_N/Z_0)$ given a set of samples $W_1, \ldots, W_r$, $W_i = \mathcal{W}[\boldsymbol{x}_i]$ from the forward distribution and a set of samples $\tilde{W}_1, \ldots, \tilde{W}_r$ $\tilde{W}_j = \mathcal{W}[\tilde{\boldsymbol{x}}_j]$ from the reverse distribution can be found by solving equation (9.5) which is given by

$$\sum_{j=1}^{r} \sigma(\tilde{W}_j - C) - \sum_{i=1}^{r} \sigma(-W_i + C) = 0 \ ,$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$.

We start by recalling equation (9.3): For a weighting function $\mathcal{F}$ we have

$$\langle \mathcal{F} \rangle_{\tilde{\mathcal{P}}} = \frac{Z_0}{Z_N} \left\langle \mathcal{F} e^{-\mathcal{W}} \right\rangle_{\mathcal{P}} \ , \tag{9.9}$$

where $\mathcal{P}$ and $\tilde{\mathcal{P}}$ denote the forward distribution and the reverse distribution of $\boldsymbol{X}$, respectively.

To take into account that a sample $\boldsymbol{x}$ of $\boldsymbol{X}$ may be either be drawn from $\mathcal{P}$ or from $\tilde{\mathcal{P}}$, we introduce an additional binary random variable $D \in \{F, R\}$ indicating if the former or the latter has been the case.

Now let us change our point of view and do not consider sampling the random variable $\boldsymbol{X}$ directly, but instead the process of sampling a certain value $W = \mathcal{W}[\boldsymbol{x}]$ of $\mathcal{W}[\boldsymbol{X}]$ where $\boldsymbol{x}$ is drawn from a mixture distribution $\mathcal{P}[\boldsymbol{x}]P(D = F) + \tilde{\mathcal{P}}[\boldsymbol{x}]P(D = R)$. For a given W we can now ask for the probabilities that it was either acquired by sampling $\boldsymbol{x}$ from the forward or the reverse distribution, that is, for the probabilities $P(F|W)$ and $P(R|W)$.

In the next step, we will derive $P(F|W)$ from equation (9.9). Let us in the following assume $P(D = F) = P(D = R) = \frac{1}{2}$ to simplify the derivation. By setting $\mathcal{F}[\boldsymbol{x}] = \delta(\mathcal{W}[\boldsymbol{x}] - W)$ where δ is the dirac-delta function, $\langle \mathcal{F} \rangle_{\mathcal{P}}$ now gives the probability $P(W|F)$ of sampling W from the forward distribution. Analogously we get $P(W|R) = \langle \mathcal{F} \rangle_{\tilde{P}}$. Inserting everything into equation (9.9) and reordering the terms gives

$$\frac{P(W|F)}{P(W|R)} = \frac{Z_N}{Z_0} e^W \ . \tag{9.10}$$

We now use Bayes' theorem to get

$$\frac{P(W|F)}{P(W|R)} = \frac{P(F|W)P(R)}{P(R|W)P(F)} = \frac{P(F|W)}{1 - P(F|W)} \ ,$$

where we used $P(F|W) + P(R|W) = 1$ in the last step. Inserting into equation (9.10) and solving for $P(F|W)$ leads to $P(F|W) = \sigma(W + \ln(Z_N/Z_0))$. Using this we can

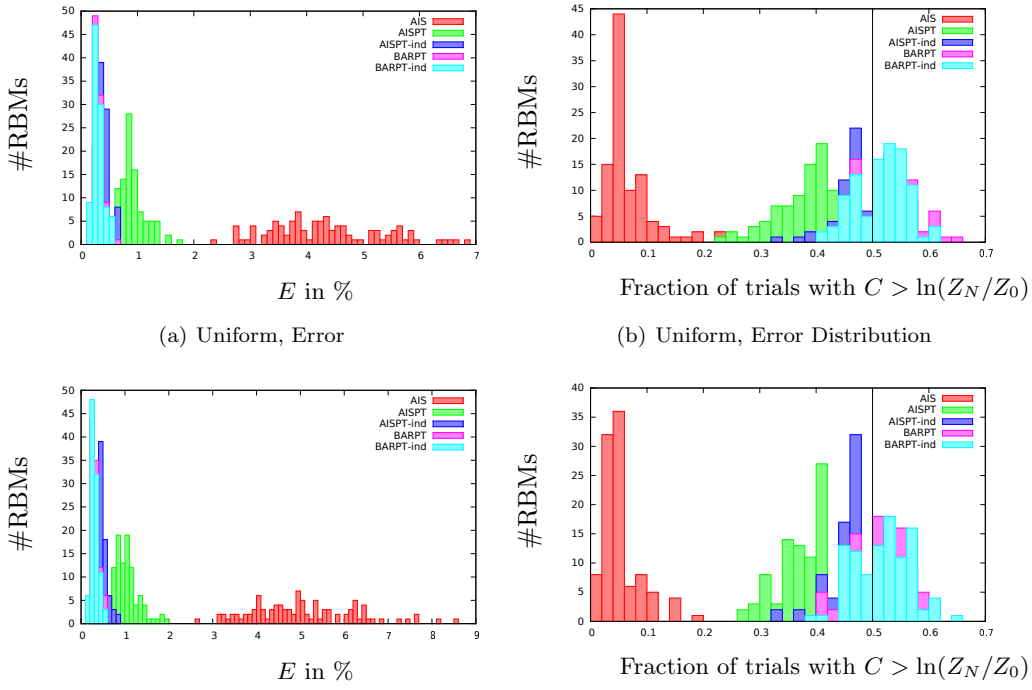
derive the probability of the reverse distribution as $P(R|W) = 1 - P(F|W) = \sigma(-W - \ln(Z_N/Z_0))$. If the quotient $C = \ln(Z_N/Z_0)$ is unknown we can treat it as a parameter and find the maximum log-likelihood solution given a set of samples $W_1, \ldots, W_r$ from the forward and a set of samples $\tilde{W}_1, \ldots, \tilde{W}_r$ from the reverse distribution. That is, we solve the optimization problem

$$\max_C \ln \left[\prod_{i=1}^{r} P(F|W_i) \prod_{j=1}^{r} P(R|\tilde{W}_j) \right] .$$

Thus, for the maximum likelihood estimate C it must hold

$$\sum_{j=1}^{r} \sigma(\tilde{W}_j + C) - \sum_{i=1}^{r} \sigma(-W_i - C) = 0 .$$

Additional experimental results



(a) Uniform, Error

(b) Uniform, Error Distribution

(c) $\alpha = 2$, Error

(d) $\alpha = 2$, Error Distribution

Figure 9.6: Repeated experiment 1 with different choices of the temperature distribution. Figures a) and b) are the corresponding results for the uniform distribution of temperatures, $\beta_i = i/N$, while c) and d) are the results for $\beta_{i,\alpha} = \frac{\sigma(\alpha i/N) - \sigma(0)}{\sigma(\alpha) - \sigma(0)}$ with $\alpha = 2$. See Figure 9.1 and the descriptions of experiment 1 and 4 for details.

Chapter 10

Approximation properties of DBNs with binary hidden units and real-valued visible units

---

This chapter is based on the manuscript "Approximation properties of DBNs with binary hidden units and real-valued visible unit" by O. Krause, A. Fischer, T. Glasmachers, and C. Igel, published in JMLR W&CP: ICML 2013, 28(1), pp. 419-426, 2013.

Abstract

Deep belief networks (DBNs) can approximate any distribution over fixed-length binary vectors. However, DBNs are frequently applied to model real-valued data, and so far little is known about their representational power in this case. We analyze the approximation properties of DBNs with two layers of binary hidden units and visible units with conditional distributions from the exponential family. It is shown that these DBNs can, under mild assumptions, model any additive mixture of distributions from the exponential family with independent variables. An arbitrarily good approximation in terms of Kullback-Leibler divergence of an m-dimensional mixture distribution with n components can be achieved by a DBN with m visible variables and n and $n+1$ hidden variables in the first and second hidden layer, respectively. Furthermore, relevant infinite mixtures can be approximated arbitrarily well by a DBN with a finite number of neurons. This includes the important special case of an infinite mixture of Gaussian distributions with fixed variance restricted to a compact domain, which in turn can approximate any strictly positive density over this domain.

10.1 Introduction

Restricted Boltzmann machines (RBMs, Smolensky, 1986; Hinton, 2002) and deep belief networks (DBNs, Hinton et al., 2006; Hinton and Salakhutdinov, 2006) are probabilistic models with latent and observable variables, which can be interpreted as stochastic neural networks. Binary RBMs, in which each variable conditioned on the others is Bernoulli distributed, are able to approximate arbitrarily well any distribution over the observable variables (Le Roux and Bengio, 2008; Montufar and Ay, 2011). Binary deep belief networks are built by layering binary RBMs, and the representational power does not decrease by adding layers (Le Roux and Bengio, 2008; Montufar and Ay, 2011). In fact, it can be shown that a binary DBN never needs more variables than a binary RBM to model a distribution with a certain accuracy (Le Roux and Bengio, 2008).

However, arguably the most prominent applications in recent times involving RBMs consider models in which the visible variables are real-valued (e.g., Salakhutdinov and Hinton, 2007; Lee et al., 2009a; Taylor et al., 2010; Le Roux et al., 2011). Welling et al. (2005) proposed a notion of RBMs where the conditional distributions of the observable variables given the latent variables and vice versa are (almost) arbitrarily chosen from the exponential family. This includes the important special case of the Gaussian-binary RBM (GB-RBM, also Gaussian-Bernoulli RBM), an RBM with binary hidden and Gaussian visible variables.

Despite their frequent use, little is known about the approximation capabilities of RBMs and DBNs modeling continuous distributions. Clearly, orchestrating a set of Bernoulli distributions to model a distribution over binary vectors is easy compared to approximating distributions over $\Omega \subseteq \mathbb{R}^m$. Recently, Wang et al. (2012) have emphasized that the distribution of the visible variables represented by a GB-RBM with n hidden units is a mixture of 2^n Gaussian distributions with means lying on the vertices of a projected n-dimensional hyperparallelotope. This limited flexibility makes modeling even a mixture of a finite number of Gaussian distributions with a GB-RBM difficult.

This work is a first step towards understanding the representational power of DBNs with binary latent and real-valued visible variables. We will show for a subset of distributions relevant in practice that DBNs with two layers of binary hidden units and a fixed family of conditional distribution for the visible units can model finite mixtures of that family arbitrarily well. As this also holds for infinite mixtures of Gaussians with fixed variance restricted to a compact domain, our results imply universal approximation of strictly positive densities over compact sets.

10.2 Background

This section will recall basic results on approximation properties of mixture distributions and binary RBMs. Furthermore, the considered models will be defined.

10.2.1 Mixture distributions

A mixture distribution $p_{\text{mix}}(\boldsymbol{v})$ over Ω is a convex combination of simpler distributions which are members of some family G of distributions over Ω parameterized by $\boldsymbol{\theta} \in \Theta$. We define $\text{MIX}(n, G) = \{\sum_{i=1}^{n} p_{\text{mix}}(\boldsymbol{v}|i)p_{\text{mix}}(i) \mid \sum_{i=1}^{n} p_{\text{mix}}(i) = 1 \text{ and } \forall i \in \{1, \ldots, n\} : p_{\text{mix}}(i) \geq 0 \wedge p_{\text{mix}}(\boldsymbol{v}|i) \in G\}$ as the family of mixtures of n distributions from G. Furthermore, we denote the family of infinite mixtures of distributions from G as $\text{CONV}(G) = \{\int_{\Theta} p(\boldsymbol{v}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \mid \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} = 1 \text{ and } \forall \boldsymbol{\theta} \in \Theta : p(\boldsymbol{\theta}) \geq 0 \wedge p(\boldsymbol{v}|\boldsymbol{\theta}) \in G\}$.

Li and Barron have shown that for some family of distributions G every element from $\text{CONV}(G)$ can be approximated arbitrarily well by finite mixtures with respect to the Kullback-Leibler divergence (KL-divergence):

Theorem 10.1 (Li and Barron, 2000)**.** *Let $f \in CONV(G)$. There exists a finite mixture $p_{mix} \in MIX(n, G)$ such that*

$$\text{KL}(f \| p_{mix}) \leq \frac{c_f^2 \gamma}{n} \ ,$$

where

$$c_f^2 = \int_{\Omega} \frac{\int f^2(\boldsymbol{v}|\boldsymbol{\theta}) f(\boldsymbol{\theta})\, d\boldsymbol{\theta}}{\int f(\boldsymbol{v}|\boldsymbol{\theta}) f(\boldsymbol{\theta})\, d\boldsymbol{\theta}}\, d\boldsymbol{v}$$

and $\gamma = 4[\log(3\sqrt{e}) + a]$ with

$$a = \sup_{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \boldsymbol{v}} \log \frac{f(\boldsymbol{v}|\boldsymbol{\theta_1})}{f(\boldsymbol{v}|\boldsymbol{\theta_2})} \ .$$

The bound is not necessarily finite. However, it follows from previous results by Zeevi and Meir (1997) that for every f and every $\epsilon > 0$ there exists a mixture p_{mix} with n components such that $\text{KL}(f \| p_{\text{mix}}) \leq \epsilon + \frac{c}{n}$ for some constant c if $\Omega \subset \mathbb{R}^m$ is a compact set and f is continuous and bounded from below by some $\eta > 0$ (i.e, $\forall x \in \Omega : f(x) \geq \eta > 0$).

Furthermore, it follows that for compact $\Omega \subset \mathbb{R}^m$ every continuous density f on Ω can be approximated arbitrarily well by an infinite but countable mixture of Gaussian distributions with fixed variance σ^2 and means restricted to Ω, that is, by a mixture of distributions from the family

$$G_{\sigma}(\Omega) = \left\{ p(\boldsymbol{x}) = \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}\|^2}{2\sigma^2} \right) \Big| \boldsymbol{x}, \boldsymbol{\mu} \in \Omega \right\} \ , \tag{10.1}$$

for sufficient small σ.

10.2.2 Restricted Boltzmann Machines

An RBM is an undirected graphical model with a bipartite structure (Smolensky, 1986; Hinton, 2002) consisting of one layer of m visible variables $\boldsymbol{V} = (V_1, \dots, V_m) \in \Omega$ and one layer of n hidden variables $\boldsymbol{H} = (H_1, \dots, H_n) \in \Lambda$. The modeled joint distribution is a Gibbs distribution $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ with energy E and normalization constant $Z = \int_\Omega \int_\Lambda e^{-E(\boldsymbol{v},\boldsymbol{h})} \,\mathrm{d}\boldsymbol{h}\,\mathrm{d}\boldsymbol{v}$, where the variables of one layer are mutually independent given the state of the other layer.

Binary-Binary-RBMs. In the standard binary RBMs the state spaces of the variables are $\Omega = \{0,1\}^m$ and $\Lambda = \{0,1\}^n$. The energy is given by $E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h} - \boldsymbol{v}^T \boldsymbol{b} - \boldsymbol{c}^T \boldsymbol{h}$ with weight matrix $\boldsymbol{W}$ and bias vectors $\boldsymbol{b}$ and $\boldsymbol{c}$.

Le Roux and Bengio showed that binary RBMs are universal approximators for distributions over binary vectors:

Theorem 10.2 (Le Roux and Bengio, 2008)**.** *Any distribution over $\Omega = \{0,1\}^m$ can be approximated arbitrarily well (with respect to the KL-divergence) with an RBM with $k+1$ hidden units, where k is the number of input vectors whose probability is not zero.*

The number of hidden neurons required can be reduced to the minimum number of pairs of input vectors differing in only one component with the property that their union contains all observable patterns having positive probability (Montufar and Ay, 2011).

Exponential-Family RBMs. Welling et al. (2005) introduced a framework for constructing generalized RBMs called exponential family harmoniums. In this framework, the conditional distributions $p(h_i|\boldsymbol{v})$ and $p(v_j|\boldsymbol{h})$, $i = 1, \dots, n$, $j = 1, \dots, m$, belong to the exponential family. Almost all types of RBMs encountered in practice, including binary RBMs, can be interpreted as exponential family harmoniums.

The exponential family is the class $\mathcal{F}$ of probability distributions that can be written in the form

$$p(\boldsymbol{x}) = \frac{1}{Z} \exp\left(\sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{x})^T \boldsymbol{\mu}^{(r)}(\boldsymbol{\theta}) \right) \quad , \tag{10.2}$$

where $\boldsymbol{\theta}$ are the parameters of the distribution and Z is the normalization constant.[1] The functions $\Phi^{(r)}$ and $\boldsymbol{\mu}^{(r)}$, for $r = 1, \dots, k$, transform the sample space and the distribution parameters, respectively. Let $\mathcal{I}$ be the subset of $\mathcal{F}$ where the components of $\boldsymbol{x} = (x_1, \dots, x_m)$ are independent from each other, that is, $\mathcal{I} = \{p \in \mathcal{F} \mid \forall \boldsymbol{x} : p(x_1, \dots, x_m) = p(x_1)p(x_2)\cdots p(x_m)\}$. For elements of $\mathcal{I}$ the function $\Phi^{(r)}$ can be

[1] By setting $k = 1$ and rewriting Φ and $\boldsymbol{\mu}$ accordingly, one obtains the standard formulation.

written as $\Phi^{(r)}(\boldsymbol{x}) = (\phi_1^{(r)}(x_1), \ldots, \phi_m^{(r)}(x_m))$. A prominent subset of $\mathcal{I}$ is the family of Gaussian distributions with fixed variance σ^2, $G_\sigma(\Omega) \subset \mathcal{I}$, see equation (10.1).

Following Welling et al., the energy of an RBM with binary hidden units and visible units with $p(\boldsymbol{v}|\boldsymbol{h}) \in \mathcal{I}$ is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \boldsymbol{W}^{(r)} \boldsymbol{h} - \sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \boldsymbol{b}^{(r)} - \boldsymbol{c}^T \boldsymbol{h} \ , \tag{10.3}$$

where $\Phi^{(r)}(\boldsymbol{v}) = (\phi_1^{(r)}(x_1), \ldots, \phi_m^{(r)}(x_m))$. Note that not every possible choice of parameters necessarily leads to a finite normalization constant and thus to a proper distribution.

If the joint distribution is properly defined, the conditional probability of the visible units given the hidden is

$$p(\boldsymbol{v}|\boldsymbol{h}) = \frac{1}{Z_{\boldsymbol{h}}} \exp\left(\sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \left(\boldsymbol{W}^{(r)} \boldsymbol{h} + \boldsymbol{b}^{(r)} \right) \right) \ , \tag{10.4}$$

where $Z_{\boldsymbol{h}}$ is the corresponding normalization constant. Thus, the marginal distribution of the visible units $p(\boldsymbol{v})$ can be expressed as a mixture of 2^n conditional distributions:

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h} \in \{0,1\}^n} p(\boldsymbol{v}|\boldsymbol{h}) p(\boldsymbol{h}) \in \mathrm{MIX}(2^n, \mathcal{I})$$

10.2.3 Deep Belief Networks

A DBN is a graphical model with more than two layers built by stacking RBMs (Hinton et al., 2006; Hinton and Salakhutdinov, 2006). A DBN with two layers of hidden variables $\boldsymbol{H}$ and $\hat{\boldsymbol{H}}$ and a visible layer $\boldsymbol{V}$ is characterized by a probability distribution $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}})$ that fulfills

$$p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) = p(\boldsymbol{v}|\boldsymbol{h}) p(\boldsymbol{h}, \hat{\boldsymbol{h}}) = p(\hat{\boldsymbol{h}}|\boldsymbol{h}) p(\boldsymbol{v}, \boldsymbol{h}) \ .$$

In this study we are interested in the approximation properties of DBNs with two binary hidden layers and real-valued visible neurons. We will refer to such a DBN as a *B-DBN*. With *B-DBN(G)* we denote the family of all B-DBNs having conditional distributions $p(\boldsymbol{v}|\boldsymbol{h}) \in G$ for all $\boldsymbol{h} \in \boldsymbol{H}$.

10.3 Approximation properties

This section will present our results on the approximation properties of DBNs with binary hidden units and real-valued visible units. It consists of the following steps:

- Lemma 10.1 gives an upper bound on the KL-divergence between a B-DBN and a finite additive mixture model – however, under the assumption that the B-DBN "contains" the mixture components. For mixture models from a subset of $\mathcal{I}$, Lemma 10.2 and Theorem 10.3 show that such B-DBNs actually exist and that the KL-divergence can be made arbitrarily small.
- Corollary 10.1 specifies the previous theorem for the special case of Gaussian mixtures, showing how the bound can be applied to distributions used in practice.
- Finally, Theorem 10.4 generalizes the results to infinite mixture distributions, and thus to the approximation of arbitrary strictly positive densities on a compact set.

10.3.1 Finite mixtures

We first introduce a construction that will enable us to model mixtures of distributions by DBNs. For some family G an arbitrary mixture of distributions $p_{\text{mix}}(\boldsymbol{v}) = \sum_{i=1}^{n} p_{\text{mix}}(\boldsymbol{v}|i)p_{\text{mix}}(i) \in \text{MIX}(n, G)$ over $\boldsymbol{v} \in \Omega$ can be expressed in terms of a joint probability distribution of $\boldsymbol{v}$ and $\boldsymbol{h} \in \{0,1\}^n$ by defining the distribution

$$q_{\text{mix}}(\boldsymbol{h}) = \begin{cases} p_{\text{mix}}(i), & \text{if } \boldsymbol{h} = \boldsymbol{e}_i, \\ 0, & \text{else} \end{cases} \tag{10.5}$$

over $\{0,1\}^n$, where $\boldsymbol{e}_i$ is the ith unit vector. Then we can rewrite $p_{\text{mix}}(\boldsymbol{v})$ as $p_{\text{mix}}(\boldsymbol{v}) = \sum_{\boldsymbol{h}} q_{\text{mix}}(\boldsymbol{v}|\boldsymbol{h})q_{\text{mix}}(\boldsymbol{h})$, where $q_{\text{mix}}(\boldsymbol{v}|\boldsymbol{h}) \in G$ for all $\boldsymbol{h} \in \{0,1\}^n$ and $q_{\text{mix}}(\boldsymbol{v}|\boldsymbol{e}_i) = p_{\text{mix}}(\boldsymbol{v}|i)$ for all $i = 1, \ldots, n$. This can be interpreted as expressing $p_{\text{mix}}(\boldsymbol{v})$ as an element of $\text{MIX}(2^n, G)$ with $2^n - n$ mixture components having a probability (or weight) equal to zero. Now we can model $p_{\text{mix}}(\boldsymbol{v})$ by the marginal distribution of the visible variables $p(\boldsymbol{v}) = \sum_{\boldsymbol{h},\hat{\boldsymbol{h}}} p(\boldsymbol{v}|\boldsymbol{h})p(\boldsymbol{h}, \hat{\boldsymbol{h}}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h})p(\boldsymbol{h})$ of a B-DBN $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) \in \text{B-DBN}(G)$ with the following properties:

1. $p(\boldsymbol{v}|\boldsymbol{e}_i) = p_{\text{mix}}(\boldsymbol{v}|i)$ for $i = 1, \ldots, n$ and
2. $p(\boldsymbol{h}) = \sum_{\hat{\boldsymbol{h}}} p(\boldsymbol{h}, \hat{\boldsymbol{h}})$ approximates $q_{\text{mix}}(\boldsymbol{h})$.

Following this line of thoughts we can formulate our first result. It provides an upper bound on the KL-divergence of any element from $\text{MIX}(n, G)$ and the marginal distribution of the visible variables of a B-DBN with the properties stated above, where $p(\boldsymbol{h})$ models $q_{\text{mix}}(\boldsymbol{h})$ with an approximation error smaller than a given ϵ.

Lemma 10.1. *Let $p_{mix}(\boldsymbol{v}) = \sum_{i=1}^{n} p_{mix}(\boldsymbol{v}|i)p_{mix}(i) \in MIX(n, G)$ be a mixture with n components from a family of distributions G, and $q_{mix}(\boldsymbol{h})$ be defined as in (10.5). Let $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) \in$ B-DBN(G) with the properties $p(\boldsymbol{v}|\boldsymbol{e}_i) = p_{mix}(\boldsymbol{v}|i)$ for $i = 1, \ldots, n$ and*

$\forall \boldsymbol{h} \in \{0,1\}^n : |p(\boldsymbol{h}) - q_{mix}(\boldsymbol{h})| < \epsilon$ *for some* $\epsilon > 0$. *Then the KL-divergence between* p_{mix} *and* p *is bounded by*

$$\mathrm{KL}(p \| p_{mix}) \leq \mathcal{B}(G, p_{mix}, \epsilon) \ ,$$

where

$$\mathcal{B}(G, p_{mix}, \epsilon) = \epsilon \int_{\Omega} \alpha(\boldsymbol{v})\beta(\boldsymbol{v}) \, d\boldsymbol{v} + 2^n(1+\epsilon)\log(1+\epsilon)$$

with

$$\alpha(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h})$$

and

$$\beta(\boldsymbol{v}) = \log\left(1 + \frac{\alpha(\boldsymbol{v})}{p_{mix}(\boldsymbol{v})}\right) \ .$$

Proof. Using $|p(\boldsymbol{h}) - q_{\mathrm{mix}}(\boldsymbol{h})| < \epsilon$ for all $\boldsymbol{h} \in \{0,1\}^n$ and $p_{\mathrm{mix}}(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h}) q_{\mathrm{mix}}(\boldsymbol{h})$ we can write

$$\begin{aligned} p(\boldsymbol{v}) &= \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h}) p(\boldsymbol{h}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h})(q_{\mathrm{mix}}(\boldsymbol{h}) + p(\boldsymbol{h}) - q_{\mathrm{mix}}(\boldsymbol{h})) \\ &= p_{\mathrm{mix}}(\boldsymbol{v}) + \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h})(p(\boldsymbol{h}) - q_{\mathrm{mix}}(\boldsymbol{h})) \leq p_{\mathrm{mix}}(\boldsymbol{v}) + \alpha(\boldsymbol{v})\epsilon \ , \end{aligned}$$

where $\alpha(\boldsymbol{v})$ is defined as above. Thus, we get for the KL-divergence

$$\begin{aligned} \mathrm{KL}(p \| p_{\mathrm{mix}}) &= \int_{\Omega} p(\boldsymbol{v}) \log\left(\frac{p(\boldsymbol{v})}{p_{\mathrm{mix}}(\boldsymbol{v})}\right) \mathrm{d}\boldsymbol{v} \\ &\leq \int_{\Omega} \underbrace{(p_{\mathrm{mix}}(\boldsymbol{v}) + \alpha(\boldsymbol{v})\epsilon) \log\left(\frac{p_{\mathrm{mix}}(\boldsymbol{v}) + \alpha(\boldsymbol{v})\epsilon}{p_{\mathrm{mix}}(\boldsymbol{v})}\right)}_{=F(\epsilon,\boldsymbol{v})} \mathrm{d}\boldsymbol{v} = \int_{\Omega} \int_0^{\epsilon} \frac{\partial}{\partial \bar{\epsilon}} F(\bar{\epsilon}, \boldsymbol{v}) \, \mathrm{d}\bar{\epsilon} \, \mathrm{d}\boldsymbol{v} \end{aligned}$$

using $F(0, \boldsymbol{v}) = 0$. Because $1 + x\epsilon \leq (1+x)(1+\epsilon)$ for all $x, \epsilon \geq 0$, we can upper bound $\frac{\partial}{\partial \epsilon} F(\epsilon, \boldsymbol{v})$ by

$$\begin{aligned} \frac{\partial}{\partial \epsilon} F(\epsilon, \boldsymbol{v}) &= \alpha(\boldsymbol{v}) \left[1 + \log\left(1 + \frac{\alpha(\boldsymbol{v})}{p_{\mathrm{mix}}(\boldsymbol{v})}\epsilon\right)\right] \\ &\leq \alpha(\boldsymbol{v}) \left[1 + \log\left((1 + \frac{\alpha(\boldsymbol{v})}{p_{\mathrm{mix}}(\boldsymbol{v})})(1+\epsilon)\right)\right] = \alpha(\boldsymbol{v}) \left[1 + \beta(\boldsymbol{v}) + \log(1+\epsilon)\right] \end{aligned}$$

with $\beta(\boldsymbol{v})$ as defined above. By integration we get

$$F(\epsilon, \boldsymbol{v}) = \int_0^{\epsilon} \frac{\partial}{\partial \bar{\epsilon}} F(\bar{\epsilon}, \boldsymbol{v}) \, \mathrm{d}\bar{\epsilon} \leq \alpha(\boldsymbol{v})\beta(\boldsymbol{v})\epsilon + \alpha(\boldsymbol{v})(1+\epsilon)\log(1+\epsilon) \ .$$

Integration with respect to $\boldsymbol{v}$ completes the proof. □

The proof does not use the independence properties of $p(\boldsymbol{v}|\boldsymbol{h})$. Thus, it is possible to apply this bound also to mixture distributions which do not have conditionally independent variables. However, in this case one has to show that a generalization of the B-DBN exists which can model the target distribution, as the formalism introduced in formula (10.3) does not cover distributions which are not in $\mathcal{I}$.

For a family $G \subseteq \mathcal{I}$ it is possible to construct a B-DBN with the properties required in Lemma 10.1 under weak technical assumptions. The assumptions hold for families of distributions used in practice, for instance Gaussian and truncated exponential distributions.

Lemma 10.2. *Let $G \subset \mathcal{I}$ and $p_{mix}(\boldsymbol{v}) = \sum_{i=1}^{n} p_{mix}(\boldsymbol{v}|i)p_{mix}(i) \in MIX(n, G)$ with*

$$p_{mix}(\boldsymbol{v}|i) = \frac{1}{Z_i} \exp\left(\sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \boldsymbol{\mu}^{(r)}(\boldsymbol{\theta}^{(i)})\right) \tag{10.6}$$

for $i = 1, \ldots, n$ and corresponding parameters $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(n)}$. Let the distribution $q_{mix}(\boldsymbol{h})$ be defined by equation (10.5). Assume that there exist parameters $\boldsymbol{b}^{(r)}$ such that for all $\boldsymbol{c} \in \mathbb{R}^n$ the joint distribution $p(\boldsymbol{v}, \boldsymbol{h})$ of $\boldsymbol{v} \in \mathbb{R}^m$ and $\boldsymbol{h} \in \{0,1\}^n$ with energy

$$E(\boldsymbol{v}, \boldsymbol{h}) = \sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \left(\boldsymbol{W}^{(r)}\boldsymbol{h} + \boldsymbol{b}^{(r)}\right) + \boldsymbol{c}^T\boldsymbol{h}$$

is a proper distribution (i.e., the corresponding normalization constant is finite), where the ith column of $\boldsymbol{W}^{(r)}$ is $\boldsymbol{\mu}^{(r)}(\boldsymbol{\theta}^{(i)}) - \boldsymbol{b}^{(r)}$. Then the following holds:

For all $\epsilon > 0$ there exists a B-DBN with joint distribution $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) = p(\boldsymbol{v}|\boldsymbol{h})p(\boldsymbol{h}, \hat{\boldsymbol{h}}) \in$ B-DBN(G) such that

i) $p_{mix}(\boldsymbol{v}|i) = p(\boldsymbol{v}|\boldsymbol{e}_i)$ for $i = 1, \ldots, n$ and

ii) $\forall \boldsymbol{h} \in \{0,1\}^n : |p(\boldsymbol{h}) - q_{mix}(\boldsymbol{h})| < \epsilon$.

Proof. Property i) follows from equation (10.4) by setting $\boldsymbol{h} = \boldsymbol{e}_i$ and the ith column of $\boldsymbol{W}^{(r)}$ to $\boldsymbol{\mu}^{(r)}(\boldsymbol{\theta}^{(i)}) - \boldsymbol{b}^{(r)}$. Property ii) follows directly from applying Theorem 10.2 to p.

□

For some families of distributions, such as truncated exponential or Gaussian distributions with uniform variance, choosing $\boldsymbol{b}^{(r)} = \boldsymbol{0}$ for $r = 1, \ldots, k$ is sufficient to yield a proper joint distribution $p(\boldsymbol{v}, \boldsymbol{h})$ and thus a B-DBN with the desired properties. If such a B-DBM exists, one can show, under weak additional assumptions on $G \subset \mathcal{I}$, that the bound shown in Lemma 10.1 is finite. It follows that the bound decreases to zero as ϵ does.

Theorem 10.3. *Let $G \subset \mathcal{I}$ be a family of densities and $p_{mix}(\boldsymbol{v}) = \sum_{i=1}^{n} p_{mix}(\boldsymbol{v}|i)p_{mix}(i) \in MIX(n, G)$ with $p_{mix}(\boldsymbol{v}|i)$ given by equation (10.6). Furthermore, let $q_{mix}(\boldsymbol{h})$ be given by equation (10.5) and let $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) \in$ B-DBN(G) with*

(i) $p_{mix}(\boldsymbol{v}|i) = p(\boldsymbol{v}|\boldsymbol{e}_i)$ *for* $i = 1, \ldots, n$

(ii) $\forall \boldsymbol{h} \in \{0,1\}^n : |p(\boldsymbol{h}) - q_{mix}(\boldsymbol{h})| < \epsilon$

(iii) $\forall \boldsymbol{h} \in \{0,1\}^n : \int\limits_{\Omega} p(\boldsymbol{v}|\boldsymbol{h}) \|\Phi^{(r)}(\boldsymbol{v})\|_1 \, d\boldsymbol{v} < \infty.$

Then $\mathcal{B}(G, p_{mix}, \epsilon)$ is finite and thus in $O(\epsilon)$.

Proof. We have to show that under the conditions given above $\int_\Omega \alpha(\boldsymbol{v})\beta(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{v}$ is finite.

We will first find an upper bound for $\beta(\boldsymbol{v}) = \log\left(1 + \frac{\alpha(\boldsymbol{v})}{p_{\mathrm{mix}}(\boldsymbol{v})}\right)$ for an arbitrary but fixed $\boldsymbol{v}$. Since $p_{\mathrm{mix}}(\boldsymbol{v}) = \sum\limits_i p_{\mathrm{mix}}(\boldsymbol{v}|i)p_{\mathrm{mix}}(i)$ is a convex combination, by defining $i^* = \arg\min_i p_{\mathrm{mix}}(\boldsymbol{v}|i)$ and $\boldsymbol{h}^* = \arg\max_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h})$ we get

$$\frac{\alpha(\boldsymbol{v})}{p_{\mathrm{mix}}(\boldsymbol{v})} = \sum_{\boldsymbol{h}} \frac{p(\boldsymbol{v}|\boldsymbol{h})}{\sum\limits_i p_{\mathrm{mix}}(\boldsymbol{v}|i)p_{\mathrm{mix}}(i)} \leq 2^n \frac{p(\boldsymbol{v}|\boldsymbol{h}^*)}{p_{\mathrm{mix}}(\boldsymbol{v}|i^*)} \quad . \tag{10.7}$$

The conditional distribution $p_{\mathrm{mix}}(\boldsymbol{v}|i)$ of the mixture can be written as in equation (10.6) and the conditional distribution $p(\boldsymbol{v}|\boldsymbol{h})$ of the RBM can be written as in formula (10.4). We define

$$\boldsymbol{u}^{(r)}(\boldsymbol{h}) = \boldsymbol{W}^{(r)}\boldsymbol{h} + \boldsymbol{b}^{(r)}$$

and get

$$\begin{aligned}
\frac{p(\boldsymbol{v}|\boldsymbol{h}^*)}{p_{\mathrm{mix}}(\boldsymbol{v}|i^*)} &= \frac{\exp\left(\sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \boldsymbol{u}^{(r)}(\boldsymbol{h}^*)\right)}{\exp\left(\sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \boldsymbol{\mu}^{(r)}(\boldsymbol{\theta}^{(i^*)})\right)} \\
&= \exp\left(\sum_{r=1}^{k} \Phi^{(r)}(\boldsymbol{v})^T \left[\boldsymbol{u}^{(r)}(\boldsymbol{h}^*) - \boldsymbol{\mu}^{(r)}(\boldsymbol{\theta}^{(i^*)})\right]\right) \\
&\leq \exp\left(\sum_{r=1}^{k}\sum_{j=1}^{m} \left|\phi_j^{(r)}(\boldsymbol{v})\right| \cdot \left|u_j^{(r)}(\boldsymbol{h}^*) - \mu_j^{(r)}(\boldsymbol{\theta}^{(i^*)})\right|\right) \quad .
\end{aligned}$$

Note that the last expression is always larger or equal to one. We can further bound this term by defining

$$\xi^{(r)} = \max_{j,\boldsymbol{h},i} \left|u_j^{(r)}(\boldsymbol{h}^*) - \mu_j^{(r)}(\boldsymbol{\theta}^{(i)})\right|$$

and arrive at

$$\frac{p(\boldsymbol{v}|\boldsymbol{h}^*)}{p_{\mathrm{mix}}(\boldsymbol{v}|i^*)} \leq \exp\left(\sum_{r=1}^{k} \xi^{(r)} \|\Phi^{(r)}(\boldsymbol{v})\|_1\right) \quad . \tag{10.8}$$

By plugging these results into the formula for $\beta(\boldsymbol{v})$ we obtain

$$\beta(\boldsymbol{v}) \overset{(10.7)}{\leq} \log\left[1 + 2^n \frac{p(\boldsymbol{v}|\boldsymbol{h}^*)}{p_{\text{mix}}(\boldsymbol{v}|i^*)}\right] \overset{(10.8)}{\leq} \log\left[1 + 2^n \exp\left(\sum_{r=1}^{k} \xi^{(r)} \|\Phi^{(r)}(\boldsymbol{v})\|_1\right)\right]$$
$$\leq \log\left[2^{n+1} \exp\left(\sum_{r=1}^{k} \xi^{(r)} \|\Phi^{(r)}(\boldsymbol{v})\|_1\right)\right] = (n+1)\log(2) + \sum_{r=1}^{k} \xi^{(r)} \|\Phi^{(r)}(\boldsymbol{v})\|_1 \ .$$

In the third step, we used that the second term is always larger than 1. Insertion into $\int_\Omega \alpha(\boldsymbol{v})\beta(\boldsymbol{v})\,\mathrm{d}\boldsymbol{v}$ leads to

$$\int_\Omega \alpha(\boldsymbol{v})\beta(\boldsymbol{v})\,\mathrm{d}\boldsymbol{v} \leq \int_\Omega \alpha(\boldsymbol{v}) \left[(n+1)\log(2) + \sum_{r=1}^{k} \xi^{(r)} \|\Phi^{(r)}(\boldsymbol{v})\|_1\right] \mathrm{d}\boldsymbol{v}$$
$$= 2^n(n+1)\log(2) + \sum_{\boldsymbol{h}} \sum_{r=1}^{k} \xi^{(r)} \int_\Omega p(\boldsymbol{v}|\boldsymbol{h}) \|\Phi^{(r)}(\boldsymbol{v})\|_1 \,\mathrm{d}\boldsymbol{v} \ , \quad (10.9)$$

which is finite by assumption. □

10.3.2 Finite Gaussian mixtures

Now we apply Lemma 10.2 and Theorem 10.3 to mixtures of Gaussian distributions with uniform variance.

The KL-divergence is continuous for strictly positive distributions. Our previous results thus imply that for every mixture p_{mix} of Gaussian distributions with uniform variance and every $\delta \geq 0$ we can find a B-DBN p such that $\mathrm{KL}(p\|p_{\text{mix}}) \leq \delta$. The following corollary gives a corresponding bound:

Corollary 10.1. *Let $\Omega = \mathbb{R}^m$ and $G_\sigma(\Omega)$ be the family of Gaussian distributions with variance σ^2. Let $\epsilon > 0$ and $p_{mix}(\boldsymbol{v}) = \sum_{i=1}^n p_{mix}(\boldsymbol{v}|i)p_{mix}(i) \in MIX(n, G_\sigma(\Omega))$ a mixture of n distributions with means $\boldsymbol{z}^{(i)} \in \mathbb{R}^m$, $i = 1, \ldots, n$. By*

$$D = \max_{\substack{r,s\in\{1,\ldots,n\}\\ k\in\{1,\ldots,m\}}} \left\{\left|z_k^{(r)} - z_k^{(s)}\right|\right\}$$

we denote the edge length of the smallest hypercube containing all means. Then there exists $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) \in$ B-DBN$(G_\sigma(\Omega))$, with $\forall \boldsymbol{h} \in \{0,1\}^n : |p(\boldsymbol{h}) - q_{mix}(\boldsymbol{h})| < \epsilon$ and $p_{mix}(\boldsymbol{v}|i) = p(\boldsymbol{v}|\boldsymbol{e}_i)$, $i = 1, \ldots, n$, such that

$$\mathrm{KL}(p\|p_{mix}) \leq \epsilon \cdot 2^n \left((n+1)\log(2) + m\left(\frac{n^2}{(\sigma/D)^2} + \frac{\sqrt{2}n}{\sqrt{\pi}(\sigma/D)}\right)\right)$$
$$+ 2^n(1+\epsilon)\log(1+\epsilon) \ .$$

Proof. In a first step we apply an affine linear transformation to map the hypercube of edge length D to the unit hypercube $[0,1]^m$. Note that doing this while transforming

the B-DBN-distribution accordingly does not change the KL-divergence, but it does change the standard deviation of the Gaussians from σ to σ/D. In other words, it suffices to show the above bound for $D = 1$ and $\boldsymbol{z}^{(i)} \in [0,1]^m$.

The energy of the Gaussian-Binary-RBM $p(\boldsymbol{v}, \boldsymbol{h})$ is typically written as

$$E(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{2\sigma^2}\boldsymbol{v}^T\boldsymbol{v} - \frac{1}{\sigma^2}\boldsymbol{v}^T\boldsymbol{b} - \frac{1}{\sigma^2}\boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h} - \boldsymbol{c}^T\boldsymbol{h} \ ,$$

with weight matrix $\boldsymbol{W}$ and bias vectors $\boldsymbol{b}$ and $\boldsymbol{c}$. This can be brought into the form of formula (10.3) by setting $k = 2$, $\phi_j^{(1)}(v_j) = v_j$, $\phi_j^{(2)}(v_j) = v_j^2$, $\boldsymbol{W}^{(1)} = \boldsymbol{W}/\sigma^2$, $\boldsymbol{W}^{(2)} = \boldsymbol{0}$, $b_j^{(1)} = b_j/\sigma^2$, and $b_j^{(2)} = 1/2\sigma^2$. With $\boldsymbol{b} = \boldsymbol{0}$ (and thus $\boldsymbol{b}^{(1)} = \boldsymbol{0}$), it follows from Lemma 10.2 that a B-DBN $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) = p(\boldsymbol{v}|\boldsymbol{h})p(\boldsymbol{h}, \hat{\boldsymbol{h}})$ with properties (i) and (ii) from Theorem 10.3 exists.

It remains to show that property (iii) holds. Since the conditional probability factorizes, it suffices to show that (iii) holds for every visible variable individually. The conditional probability of the jth visible neuron of the constructed B-DBN is given by

$$p(v_j|\boldsymbol{h}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(v_j - z_j(\boldsymbol{h}))^2}{2\sigma^2}\right) \ ,$$

where the mean $z_j(\boldsymbol{h})$ is the jth element of $\boldsymbol{W}\boldsymbol{h}$. Using this, it is easy to see that

$$\int_{-\infty}^{\infty} p(v_j|\boldsymbol{h})|\phi^{(2)}(v_j)|\,\mathrm{d}v_j = \int_{-\infty}^{\infty} p(v_j|\boldsymbol{h})v_j^2\,\mathrm{d}v_j < \infty \ ,$$

because it is the second moment of the normal distribution. For $\int_{-\infty}^{\infty} p(v_j|\boldsymbol{h})|\phi^{(1)}(v_j)|\,\mathrm{d}v_j$ we get

$$\begin{aligned}
&\frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{(v_j - z_j(\boldsymbol{h}))^2}{2\sigma^2}\right)|v_j|\,\mathrm{d}v_j \\
&\qquad = -z_j(\boldsymbol{h}) + \frac{2}{\sqrt{2\pi\sigma^2}} \int_0^{\infty} \exp\left(-\frac{(v_j - z_j(\boldsymbol{h}))^2}{2\sigma^2}\right) v_j\,\mathrm{d}v_j \\
&\qquad = -z_j(\boldsymbol{h}) + \frac{2}{\sqrt{2\pi\sigma^2}} \int_{-z_j(\boldsymbol{h})}^{\infty} \exp\left(-\frac{t^2}{2\sigma^2}\right)(t + z_j(\boldsymbol{h}))\mathrm{d}t \\
&= -z_j(\boldsymbol{h}) + 2z_j(\boldsymbol{h}) \int_0^{\infty} p(v_j|\boldsymbol{h})\mathrm{d}v_j + \frac{2}{\sqrt{2\pi\sigma^2}} \int_{-z_j(\boldsymbol{h})}^{\infty} \exp\left(-\frac{t^2}{2\sigma^2}\right) t\,\mathrm{d}t \\
&\qquad \leq z_j(\boldsymbol{h}) + \frac{\sqrt{2}\sigma}{\sqrt{\pi}} \exp\left(-\frac{z_j^2(\boldsymbol{h})}{2\sigma^2}\right) \leq n + \frac{\sqrt{2}\sigma}{\sqrt{\pi}} \ . \qquad (10.10)
\end{aligned}$$

In the last step we used that $z_j(\boldsymbol{e}_i) = z_j^{(i)} \in [0,1]$ by construction and thus $z_j(\boldsymbol{h})$ can be bounded from above by

$$z_j(\boldsymbol{h}) = \sum_{i=0}^{n} h_i z_j(\boldsymbol{e}_i) \leq n \ . \qquad (10.11)$$

Thus it follows from Theorem 10.3 that the bound from Lemma 10.1 holds and is finite. To get the actual bound, we only need to find the constants $\xi^{(1)}$ and $\xi^{(2)}$ to be inserted

into (10.9). The first constant is given by $\xi^{(1)} = \max_{j,\boldsymbol{h},i} \left| \frac{z_j(\boldsymbol{h})}{\sigma^2} - \frac{z_j^{(i)}}{\sigma^2} \right|$. It can be upper bounded by $\max_{j,\boldsymbol{h}} \frac{z_j(\boldsymbol{h})}{\sigma^2} \leq \frac{n}{\sigma^2}$, as an application of equation (10.11) shows. The second constant is given by $\xi^{(2)} = \max_{j,\boldsymbol{h},i} |\frac{1}{2\sigma^2} - \frac{1}{2\sigma^2}| = 0$. Inserting these variables into inequality (10.9) leads to the bound. □

The bound $\mathcal{B}(G_\sigma(\Omega), p_{\text{mix}}, \epsilon)$ is also finite when Ω is restricted to a compact subset of $\mathbb{R}^m$. This can easily be verified by adapting equation (10.10) accordingly.

Similar results can be obtained for other families of distributions. A prominent example are B-DBMs with truncated exponential distributions. In this case the energy function of the first layer is the same as for the binary RBM, but the values of the visible neurons are chosen from the interval $[0, 1]$ instead of $\{0, 1\}$. It is easy to see that for every choice of parameters the normalization constant as well as the bound are finite.

10.3.3 Infinite mixtures

We will now transfer our results for finite mixtures to the case of infinite mixtures following Li and Barron (2000).

Theorem 10.4. *Let G be a family of continuous distributions and $f \in CONV(G)$ such that the bound from Theorem 10.1 is finite for all $p_{mix\text{-}n} \in MIX(n, G)$, $n \in \mathbb{N}$. Furthermore, for all $p_{mix\text{-}n} \in MIX(n, G)$, $n \in \mathbb{N}$, and for all $\hat{\epsilon} > 0$ let there exist a B-DBN in B-DBN(G) such that $\mathcal{B}(G, p_{mix}, \hat{\epsilon})$ is finite. Then for all $\epsilon > 0$ there exists $p(\boldsymbol{v}, \boldsymbol{h}, \hat{\boldsymbol{h}}) \in$ B-DBN(G) with $KL(f\|p) \leq \epsilon$.*

Proof. From Theorem 10.1 and the assumption that the corresponding bound is finite it follows that for all $\epsilon > 0$ there exists a mixture $p_{\text{mix-}n'} \in \text{MIX}(n', G)$ with $n' \geq 2c_f^2\gamma/\epsilon$ such that $\text{KL}(f\|p_{\text{mix-}n'}) \leq \frac{\epsilon}{2}$.

By assumption there exists a B-DBN $\in$ B-DBN(G) such that $\mathcal{B}(G, p_{\text{mix-}n'}, \hat{\epsilon})$ is finite. Thus, one can define a sequence of B-DBNs $(p_{\hat{\epsilon}})_{\hat{\epsilon}} \in$ B-DBN(G) with $\hat{\epsilon}$ decaying to zero (where the B-DBNs only differ in the weights between the hidden layers) for which it holds $\text{KL}(p_{\hat{\epsilon}}\|p_{\text{mix-}n'}) \overset{\hat{\epsilon}\to 0}{\longrightarrow} 0$. This implies that $p_{\hat{\epsilon}} \overset{\hat{\epsilon}\to 0}{\longrightarrow} p_{\text{mix-}n'}$ uniformly. It follows $\text{KL}(f\|p_{\hat{\epsilon}}) \overset{\hat{\epsilon}\to 0}{\longrightarrow} \text{KL}(f\|p_{\text{mix-}n'})$. Thus, there exists ϵ' such that $|\text{KL}(f\|p_{\epsilon'}) - \text{KL}(f\|p_{\text{mix-}n'})| < \epsilon/2$. A combination of these inequalities yields

$$\text{KL}(f\|p_{\epsilon'}) \quad \leq \quad |\,\text{KL}(f\|p_{\epsilon'}) \;-\; \text{KL}(f\|p_{\text{mix-}n'})| \;+\; \text{KL}(f\|p_{\text{mix-}n'}) \quad \leq \quad \epsilon\ .$$

□

This result applies to infinite mixtures of Gaussians with the same fixed but arbitrary variance σ^2 in all components. In the limit $\sigma \to 0$ such mixtures can approximate strictly positive densities over compact sets arbitrarily well (Zeevi and Meir, 1997).

10.4 Conclusions

We presented a step towards understanding the representational power of DBNs for modeling real-valued data. When binary latent variables are considered, DBNs with two hidden layers can already achieve good approximation results. Under mild constraints, we showed that for modeling a mixture of n pairwise independent distributions, a DBN with only $2n + 1$ binary hidden units is sufficient to make the KL-divergence between the mixture p_{mix} and the DBN distribution p arbitrarily small (i.e., for every $\delta > 0$ we can find a DBN such that $\text{KL}(p\|p_{\text{mix}}) < \delta$). This holds for deep architectures used in practice, for instance DBNs having visible neurons with Gaussian or truncated exponential conditional distributions, and corresponding mixture distributions having components of the same type as the visible units of the DBN. Furthermore, we extended these results to infinite mixtures and showed that these can be approximated arbitrarily well by a DBN with a finite number of neurons. Therefore, Gaussian-binary DBNs inherit the universal approximation properties from additive Gaussian mixtures, which can model any strictly positive density over a compact domain with arbitrarily high accuracy.

Chapter 11

Discussion and conclusion

This thesis presented a series of research articles addressing challenges and hitherto open questions in the context of the training of Restricted Boltzmann Machines (RBMs). The first set of articles analyzed different RBM training algorithms; the second set presented different approaches to improve learning; and the last article analyzed the representational power of Deep Belief Networks (DBNs) with real-valued visible variables. In the following the main results will be summarized and discussed.

11.1 Summary and discussion

Restricted Boltzmann machines are Markov random fields also known as undirected graphical models. The training of RBMs is based on gradient ascent on Markov Chain Monte Carlo (MCMC) based approximations of the log-likelihood gradient. Despite the existence of a number of sound tutorials which cover RBM training (e.g., Bengio, 2009; Swersky et al., 2010; Hinton, 2012), a self-contained introduction to RBMs from a statistical perspective was missing so far. Therefore, Chapter 2 presented an introductory tutorial on training RBMs embedding them into the framework of probabilistic graphical models and providing the required concepts from Markov chain theory.

Training undirected graphical models such as RBMs is challenging. The training is based on likelihood maximization, but the likelihood and its gradient are intractable. This is due to a normalization constant involving a number of terms that grows exponentially with the size of the model. Getting unbiased approximations of the gradient by MCMC methods typically needs too many sampling steps to be computationally efficient. Learning algorithms for RBMs, such as k-step Contrastive Divergence (CD, Hinton, 2002) or (Fast) Persistent CD ((F)PCD, Tieleman, 2008; Tieleman and Hinton, 2009), make use of the fact that a gradient approximation based on samples from

a Gibbs chain iterated only for a small number k of steps (and usually $k = 1$) appears to be sufficient for training. Obviously the resulting approximations are biased. As follows from Markov chain theory, this bias depends on k and the mixing rate of the Gibbs chain, and it is known that the mixing rate decreases with increasing magnitude of the RBM parameters (Hinton, 2002; Carreira-Perpiñán and Hinton, 2005; Bengio and Delalleau, 2009).

11.1.1 An analysis of RBM training algorithms

The first part of the thesis analyzed different aspects of RBM training algorithms.

An analysis of the approximation bias of Gibbs sampling based training methods. Chapter 3 empirically analyzed the impact of the bias of the Gibbs sampling based gradient approximations used in CD, PCD and FPCD learning. While it is common knowledge that learning based on the biased approximations may only result in an approximation of a maximum likelihood solution (Carreira-Perpiñán and Hinton, 2005; Bengio and Delalleau, 2009), it was shown that bias can even lead to a distortion of the learning process: after an initial increase, the likelihood can start to diverge, and thus the bias can lead to a systematic and drastic decrease of model quality.[1] This can be explained by the increase of the absolute values of the model parameters during learning that steadily slows down the mixing rate of the Gibbs chain associated with the gradient approximation.

In CD learning the Gibbs chain is initialized with a sample from the training set and the modeled distribution gets closer to this starting distribution during training. Nevertheless, the property of Gibbs chains to generally converge faster if the starting distribution is close to the target distribution seems not to compensate for the increase of parameter magnitudes.

In accordance with the fact that the bias decreases with increasing number of sampling steps, it was found that increasing k leads to models with higher likelihood and can prevent divergence. However, divergence occurs even for values of k too large to be computationally tractable for large models. Furthermore, the analysis showed that the divergence can be avoided by an adaptive learning rate or the usage of weight decay, though only when an appropriate annealing schedule or weight decay parameter is chosen. However, for doing so one would need a reliable heuristics for choosing the annealing schedule or the weight decay parameter. The divergence could also be avoided by early stopping, but this requires some reliable indicator that tells us when to stop, and which can be computed efficiently. However, the likelihood is only tractable

[1] The divergence effects were reported before by Fischer and Igel (2009) and Desjardins et al. (2010b) but analyzed first in detail in the presented work.

for small models and it was shown in this work that the reconstruction error, which has been suggested for monitoring the training progress (Bengio et al., 2007; Taylor et al., 2007), may further decrease in spite of a divergence of the likelihood and thus is not reliable. Here efficient estimators for the likelihood could be a solution, for example Bennett's Acceptance Ratio (BAR, Bennett, 1976), an estimation method further analyzed along with variants of Annealed Importance Sampling (AIS, Neal, 2001) in Chapter 9 of this thesis, as it will be discussed below.

It was reported by Bengio and Delalleau (2009) that, despite of the bias, the signs of most components of the CD update are equal to the corresponding signs of the log-likelihood gradient (that is, the signs of the corresponding log-likelihood derivatives). Therefore, the usage of optimization techniques only depending on the signs, such as resilient backpropagation (RProp, Riedmiller, 1994; Igel and Hüsken, 2003), seems promising. This idea was investigated in Chapter 4, where RProp was combined with CD learning for RBM training. It was found that if no divergence occurs for steepest ascent on the gradient approximation, the distributions underlying the training data could also be learned with Rprop. However, if the likelihood diverges when using steepest ascent, the divergence became even more severe when using Rprop. This is due to the faster growth of the RBM parameters induced by Rporp, which also leads to a faster increase of the approximation bias. Thus, although the sign of the components of the CD update direction vector has been reported to often be right, learning based on these signs tends to diverge.

The work in Chapter 5 theoretically analyzed the CD-k approximation bias by deriving an upper bound for the expected approximation error. It is based on a well known upper bound for the convergence rate of the Gibbs sampler (see e.g., Brémaud, 1999, highlighting again the close connection between the magnitude of the bias and the mixing rate of the Gibbs chain. The new bound is considerably tighter than a previous published result (Bengio and Delalleau, 2009). Moreover, the derived bound shows a dependency on the magnitude of the RBM parameters and the number of sampling steps that is in line with the empirical results given in Chapter 3 and discussed above. It increases with increasing absolute values of the model parameters, reflecting the dependency of the approximation bias on the parameters and indicating the relevance of controlling the absolute parameter values, for example by using weight-decay. Like the bias, the upper bound decreases with increasing number k of sampling steps emphasizing the fact that larger values of k stabilize CD learning. Furthermore, the bound increases with increasing size of the RBM (that is, with increasing number of variables) and decreases with decreasing distance between the modeled distribution and the starting distribution of the Gibbs chain.

In the presented analysis the starting distribution was chosen to be the empirical

distribution underlying the training data, since in CD learning the Gibbs chain is initialized with a sample from the training set. If the starting distribution is chosen to be the distribution given by the persistent Gibbs chain employed in PCD learning instead, the results can also be applied to bounding the approximation error of this training method.

Experiments comparing the values of bias and bound of the CD approximation for small RBMs trained on toy data sets showed the tightness of the new bound. Only in the initial phase of learning, the bound was rather loose. While the bound takes rather large values in the beginning, because the distance between starting and model distribution is large, the bias is small in the initial phase when the parameters are close to zero and Gibbs sampling mixes fast. However, the difference between bias and bound decreases fast and the bound gets tight during training.

An analysis of the mixing rate of PT sampling in RBMs. Parallel tempering (PT, Swendsen and Wang, 1986; Geyer, 1991) is an advanced sampling technique aimed at increasing the mixing rate of Metropolis-Hastings based methods such as Gibbs sampling. It samples in parallel from several tempered Gibbs chains, corresponding to more and more smoothed versions of the original chain, and allows samples to swap between chains. Parallel tempering was successfully applied for sampling in RBM training, where it was shown to lead to better generative models (Desjardins et al., 2010b; Cho et al., 2010). Furthermore, it can prevent the divergence of the log-likelihood if the number of parallel chains is sufficiently large. This can for example be seen from the results of the introductory experiments presented in Chapter 2.

Chapter 6 presented the first analysis of the convergence rate of PT for sampling in RBMs. Based on general results for the mixing rate of PT (Woodard et al., 2009b), a lower bound on the spectral gap was derived. This yielded an upper bound on the convergence rate, which shows an exponential dependency on the maximum size of the two layers and the sum of the absolute values of the RBM parameters. Thus, the bound indicates that mixing slows down with an increase of the number of variables and the magnitude of the model parameters. These dependencies are similar to those of the well known bound on the convergence rate of the Gibbs sampler (see, e.g. Brémaud, 1999) used for deriving the bound on the CD-approximation bias in Chapter 5 as discussed above.

Since the results of empirical studies imply that PT has better mixing properties than Gibbs sampling, one would like to find a bound on the convergence rate of PT that is tighter than that on the convergence rate of the Gibbs sampler. However, this property does not hold for the bound derived in this thesis. One reason for this is that it bounds the convergence of the ensemble of all tempered chains used in parallel (also

referred to as product chain) and not only of the original chain and the product chain always converges slower than the single chains. Finding a direct bound on the mixing rate of the original chain instead seems difficult. To my knowledge, all approaches to analyze the mixing rate of general PT sampling published so far consider the ensemble of chains (e.g., Woodard et al. (2009b); Madras and Zheng (2003); Bhatnagar and Randall (2004)). This also explains why they all suffer (as the derived bound) from the same linear dependency on the number of parallel chains.

The derived bound is arguably loose, but it is non trivial and presents the first approach to investigate the mixing rate of PT for RBMs. Furthermore, it may be interpreted as being in favor of the conjecture that it is not possible to get rid of the exponential dependencies of the mixing rate on the RBM complexity. This would mean that PT for RBMs is not rapidly mixing, a property shown to be true for related models from physics (as for example certain types of the mean field Potts model, or the mean field Ising model if the number of parallel chains used in PT is not increased with the model complexity (Woodard et al., 2009a)). However, this conjecture needs to be further investigated.

11.1.2 Improvements for RBM training

The second part of the thesis presented several improvements for RBM training.

A transition operator for sampling in RBMs that increases the mixing rate. The results described in the previous section emphasize that the performance of RBM learning algorithms strongly depends on the mixing rate of the Markov chains used for sampling. The sampling techniques used in CD learning and its variants as well as PT rely on Gibbs sampling, which is a Metropolis-type transition operator. Chapter 7 suggested to replace Gibbs sampling by another transition operator from this family. The proposed operator was designed to maximize the probability of state changes and is referred to as flip-the-state operator. It is related to the Metropolized Gibbs sampler previously discussed as an alternative to the standard Gibbs sampler for sampling in Ising models (Neal, 1993; Liu, 1996).

Chapter 7 presented a theoretical as well as an empirical analysis of the proposed operator. In the theoretical analysis, it was proven that the flip-the-state operator induces an aperiodic and irreducible Markov chain, which guarantees that it is properly converging to the stationary distribution. The empirical analysis compared the mixing behavior of the flip-the-state method with Gibbs sampling in various experiments investigating the second largest eigenvalues of the corresponding transition matrices, the induced autocorrelation times and the resulting number of class changes that reflect mode changes. While Gibbs sampling is optimal if the RBM parameters are (close

to) zero, the results clearly show that the proposed flip-the-state method increases the mixing rate compared to Gibbs sampling when the magnitude of the parameters increases. Better mixing properties in a scenario with large weights make the flip-the-state operator especially promising for sampling in learning methods. A comparison of the standard learning methods CD, PCD and training based on PT employing either the Gibbs or the flip-the-state transition operator indeed showed that the proposed operator leads to better learning results. Statistically significant higher likelihood values were reached during training for most experimental settings. The improvements on the learning outcome were rather small, but consistently observed for all learning algorithms. Furthermore, flip-the-state sampling does not introduce computational overhead. Therefore, it should clearly be used instead of Gibbs sampling in practice.

A way of parametrizing RBMs that leads to better models and robustness against changes of the data representation. Recently, Montavon and Müller (2012) showed that subtracting mean values from the variables of Deep Boltzmann Machines (DBMs) leads to better conditioned optimization problems and to better generative properties in the case of locally connected DBMs.

Inspired by this work, Chapter 8 analyzed centered binary RBMs, where centering corresponds to subtracting offset parameters from visible and hidden variables. It was shown analytically that, since centered RBMs and normal RBMs are different parameterizations of the same model class, training a centered RBM can be reformulated to training a normal binary RBM based on a new update direction, which is used instead of the gradient and is called the centered gradient. From this new formulation followed that the enhanced gradient (Cho et al., 2011) is equivalent to centering for a certain choice of offset parameters. The enhanced gradient was designed as an alternative update direction replacing the log-likelihood gradient in RBM training, with the aim of making the training procedure more robust against changes of the input representation. In particular, training should perform equally well on a data set and the inverted version of the same set (generated by flipping all bits). This desired invariance of the training performance to changes of the data representation holds more generally for centered RBMs for a broad set of offset values as proven in this work.

An empirical analysis showed that centered RBMs are not only robust against changes of the data representation but can also reach significantly higher log-likelihood values than normal binary RBMs. The analysis comprised a comparison of different offset values and different ways to train centered RBMs, including the centering version equal to the enhanced gradient, the paramterization subtracting the data mean from the visible variables introduced by Tang and Sutskever (2011) and the centering version suggested by Montavon and Müller (2012). The comparison showed that optimal

performance is achieved when both visible and hidden variables are centered and when the offsets are set to (approximations of) the variable expectations under the data or model distribution. However, using the expectation under the RBM distribution (as for example the enhanced gradient does) can lead to a severe divergence of the log-likelihood when using PT for training. This can be prevented when an exponentially moving average is applied to the approximations of the offset values.

One explanation for the superiority of centered RBMs is that they explicitly model the mean values, which allows the weights to model second and higher order statistics right from the start. This is in contrast to normal binary RBMs, where the weights usually capture parts of the mean values. This explanation was supported by an empirical comparison of the norms of weight and bias parameters showing that training centered RBMs leads to smaller weight norms and larger bias norms compared to normal binary RBMs. Another experiment compared the centered gradient and the standard gradient to the natural gradient (Amari, 1998), which would be the update direction of choice if it were tractable for RBMs (Desjardins et al., 2013). An investigation of the angle between the centered and the natural gradient as well as the angle between the standard and the natural gradient showed that the centered gradient is closer to the natural gradient supporting the observed superiority of centered RBMs. In summary, all presented results clearly show that centering should always be used when training binary RBMs.

New estimators for the normalization constant for assessing model quality. Computing the log-likelihood of the RBM parameters given some data requires to compute the normalization constant (also referred to as partition function), which is only tractable for small RBMs. This makes it difficult to assess the model quality of trained RBMs, to monitor the training process, or to perform likelihood ratio tests. Therefore, statistical techniques for efficiently estimating the normalization constant are needed. So far, two estimation methods borrowed from statistical physics have been applied for estimating the partition function of RBMs: Salakhutdinov and Murray (2008) suggested to use AIS and Desjardins et al. (2011) employed BAR in combination with an importance sampling based estimator using samples from previous learning iterations and a Kalman filter like inference procedure.

Chapter 9 introduced a theoretic framework for deriving estimates for the fraction of the normalization constants of two distributions (where one can be chosen to be the RBM distribution and the other to be a reference distribution for which the normalization constant is known). The framework uses a generalized form of Crooks' equality (Crooks, 2000), which links this fraction of normalization constants to the fraction of two expectations, one over a distribution of samples generated by a transition op-

erator and the other over the distribution induced by the reversed operator. From there, generalizations of AIS and BAR can be derived, which allow the use of different sampling methods for drawing samples from a set of bridging distributions connecting the RBM and the reference distribution. The analysis focused on path sampling (as typically used for AIS (Neal, 2001)) and PT as methods for generating dependent or independent samples from a set of bridging distributions, respectively.

In a set of experiments, the partition function estimation via vanilla AIS was compared with AIS- and BAR-based estimators using independent samples from PT. The results showed that all approaches using PT lead to better estimation results than vanilla AIS. Furthermore, algorithms based on AIS were clearly outperformed by BAR-like estimators. This is in accordance with the results reported by Desjardins et al. (2011), who found that the RBM likelihood can efficiently be tracked with an estimation procedure using, among others, BAR and PT as components. A comparison of the estimation errors for the normalization constant of randomly generated RBMs further showed that AIS tends to underestimate the true value of the partition function, while the distribution of the approximation error of BAR was almost symmetric. The tendency of underestimation was especially strong for vanilla AIS and already reduced if AIS variants relying on PT samples were employed. Experiments varying the number of bridging distributions and the bridging distributions themselves showed the superiority of BAR over AIS especially for settings where only a small number of bridging chains are employed. Moreover, the performance of AIS strongly depends on the right choice of bridging distributions, while BAR worked reliably across a range of different distributions. The results further showed that if PT is employed for sampling from the bridging distributions, the estimation performance depends on the sample quality. Thus, it is important that the burn-in times are not too short. However, when the estimators were used to track the partition function during PT based training (reusing the samples generated for learning), the persistent PT chains seemed to be sufficiently close to the bridging distributions such that biased samples were not a problem. In summary, the results clearly suggest to use BAR with PT instead of AIS to estimate the normalization constants of RBMs.

11.1.3 An analysis of the representational power of DBNs with real-valued visible variables

Restricted Boltzmann machines are the building blocks of DBNs. While it is known that DBNs with binary variables can approximate any distribution over fixed-length binary vectors arbitrarily well (Le Roux and Bengio, 2008; Montufar and Ay, 2011; Le Roux and Bengio, 2010), little is known about the approximation capabilities of DBNs modeling distributions over real-valued data.

Chapter 10 contributed to filling this gap by analyzing the representational power of DBNs with two layers of binary hidden variables and real-valued visible variables with conditional distributions from the exponential family. It was shown that, under mild technical assumptions, these DBNs can model any additive mixture of distributions from the exponential family with independent variables arbitrarily well. This was done by deriving an upper bound on the Kullback-Leibler divergence between the model and the mixture distribution. The bound can be made arbitrarily small for an m dimensional mixture distribution of n pairwise independent components and the distribution represented by a DBN with m visible variables and n and $n+1$ hidden variables in the first and second hidden layer, respectively. The required technical assumptions hold, for example, for DBNs having visible variables with Gaussian or truncated exponential conditional distributions and corresponding mixture distributions having components of the same type. Thus, the results hold for architectures relevant in practice.

The results were further transferred from finite to infinite mixtures. It was shown that an infinite mixture can be approximated arbitrarily well by a DBN with a finite number of variables. This also applies to infinite additive mixtures of Gaussians, which in turn can model any strictly positive density over a compact domain with arbitrary high accuracy (Zeevi and Meir, 1997). Therefore, DBNs with Gaussian visible and binary hidden neurons can also model any strictly positive density over a compact domain arbitrarily well.

A similar idea as underlying this analysis was also used by Cho et al. (2013a) for proving universal approximation properties for DBMs with Gaussian visible and two layers of binary hidden variables.

11.2 Conclusion

In this thesis it was shown that the bias of Gibbs sampling based approximations of the log-likelihood gradient as used for CD or PCD learning in RBMs can lead to a divergence of the likelihood and thus to a severe disturbance of the learning process. This divergence occurs for gradient ascent as well as for Rprop, a gradient based optimization technique only relying on the signs of the derivatives. The approximation bias can be upper bounded by an expression reflecting, among others, the dependency of the approximation error on the number of sampling steps and the magnitude of the RBM parameters, which is known to influence the mixing rate of the Gibbs chain. In this thesis, the first analysis of the convergence rate of PT was presented, an advanced sampling technique aiming at increasing the mixing rate of Gibbs sampling by running several Gibbs chains in parallel, that leads to higher log-likelihood values and can prevent divergence in RBM training. Furthermore, it was shown that the mixing

rate of all sampling methods used for RBM training can be increased by replacing the Gibbs sampler by the proposed flip-the-state transition operator, that maximizes the probability of state changes. By subtracting the mean from the variables, which leads to centered RBMs (a different parametrization of the same model class), the training procedure gets more robust against changes of the data representation and better models of the training data can be learned. An analysis showed that the BAR method gives results superior to AIS for estimating the likelihood of the RBM parameters, and that it can be employed to reliably assess model performance and training progress, optimally during PT based training. Finally, the representational power of DBNs with real-valued visible variables was analyzed and it was shown that (under mild assumptions) an DBN with $2n + 1$ hidden units can model any additive mixture of n distributions from the exponential family with independent variables arbitrarily well. Furthermore, a finite number of hidden variables is sufficient to approximate any strictly positive density over a compact domain.

Summarizing the main conclusions for training RBMs in practice, I recommend training centered RBMs with PT and employing the flip-the-state operator. Furthermore BAR should be preferred over AIS for tracking the model performance.

11.3 Future work

Most of the work presented in this thesis focused on RBMs with binary variables. Some of the concepts and ideas could be transferred to RBMs with real valued variables, most prominently RBMs having binary hidden and visible variables with a Gaussian conditional distribution, also called Gaussian-Bernoulli-RBMs (GB-RBMs). Firstly, the bounds on the approximation bias of CD and the convergence rate of PT can be adapted to sampling from GB-RBMs. Secondly, the idea underlying the flip-the-state operator, namely the idea of 'trying to move away' from the current value of a variable when drawing a new value, can also be applied to RBM variables with Gaussian conditional distributions. For this purpose, the over-relaxation method described by Adler (1981) could be employed, where a new value of a variable is drawn from a Gaussian that is biased to the side of the conditional distribution opposite to the current value. How this influences the mixing behavior of sampling methods and the learning outcome of RBM training makes up an interesting question. Thirdly, analyzing centered RBMs with Gaussian variables could be interesting, since Cho et al. (2013a) reported that the enhanced gradient, which was shown to be a version of centering, improves the training of Gaussian-Bernoulli-DBMs.

Furthermore, the hypothesis that PT is not rapid but torpid mixing in RBMs, that means that the convergence rate decreases exponentially and not polynomially as a

function of the size of the RBM, needs to be further investigated. An approach could be based on the conditions for torpid mixing of PT specified by Woodard et al. (2009a).

Arguably, PT is one of the most successful methods for sampling in RBMs. In Markov random fields often used in physics, like the Ising and the Potts model, another family of MCMC techniques, sometimes referred to as cluster MCMC methods, became quite popular (Wang and Swendsen, 1990). Prominent examples are the Swendsen-Wang and the Wolff algorithm (Swendsen and Wang, 1987; Wolff, 1989). The basic idea of these type of methods is to sample a new value for a whole cluster of variables and not only for single variable in each step. It is interesting to investigate if this principle could also be applied to sampling in RBMs.

On the other hand, some of the results from this thesis may also be relevant to the field of physics. For example the convergence proof for the flip-the-state operator could be adapted to the Ising and Potts model.

Bibliography

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

S. L. Adler. Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions. *Physical Review D*, 23:2901–2904, 1981.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2): 251–276, 1998.

S. Amari, K. Koji, and N. Hiroshi. Information geometry of Boltzmann machines. *IEEE Transactions on Neural Networks*, 3(2):260–271, 1992.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 21(6):1601–1621, 2009.

Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. 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.

Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *JMLR W&CP*, pages 552–560, 2013.

C. H. Bennett. Efficient estimation of free energy differences from Monte Carlo data. *Journal of Computational Physics*, 22(2):245 – 268, 1976.

N. Bhatnagar and D. Randall. Torpid mixing of simulated tempering on the potts model. In J. I. Munro, editor, *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 478–487. SIAM, 2004.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

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.

P. Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues.* Springer-Verlag, 1999.

O. Breuleux, Y. Bengio, and P. Vincent. Quickly generating representative samples from an RBM-derived process. *Neural computation*, 23(8):2058–2073, 2011.

K. Brügge, A. Fischer, and C. Igel. The flip-the-state transition operator for restricted Boltzmann machines. *Machine Learning*, 13:53–69, 2013.

S. Caracciolo, A. Pelissetto, and A. D. Sokal. Two remarks on simulated tempering. *Unpublished manuscript*, 1992.

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.

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.

K. Cho, T. Raiko, and A. Ilin. Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In L. Getoor and T. Scheffer, editors, *Proceedings of 28th International Conference on Machine Learning (ICML)*, pages 105–112. ACM, 2011.

K. Cho, T. Raiko, and A. Ilin. Gaussian-Bernoulli deep Boltzmann machines. In *In Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013a.

K. Cho, T. Raiko, and A. Ilin. Enhanced gradient for training restricted Boltzmann machines. *Neural Computation*, 25:805–831, 2013b.

A. Courville, J. Bergstra, and Y. Bengio. Unsupervised models of images by spike-and-slab RBMs. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1145–1152. ACM, 2011.

G. E. Crooks. Path-ensemble averages in systems driven far from equilibrium. *Physical Review E*, 61:2361–2366, 2000.

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*, 2010a.

G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Dellaleau. Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines. In Y. W. Teh and M. Titterington, editors, *Proceedings of the 13th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR W&CP*, pages 145–152, 2010b.

G. Desjardins, A. C. Courville, and Y. Bengio. On tracking the partition function. In *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 2501–2509. MIT Press, 2011.

G. Desjardins, R. Pascanu, A. Courville, and Y. Bengio. Metric-free natural gradient for joint-training of Boltzmann machines. CoRR, abs/arXiv:1301.3545, 2013.

P. Diaconis and L. Saloff-Coste. Comparison theorems for reversible Markov chains. *The Annals of Applied Probability*, 3:696–730, 1993.

P. Diaconis and L. Saloff-Coste. Logarithmic Sobolev inequalities for finite Markov chains. *The Annals of Applied Probability*, 6:695–750, 1996.

P. Diaconis and L. Saloff-Coste. What do we know about the Metropolis algorithm? *Journal of Computer and System Sciences*, 57:20–36, 1998.

D. Erhan, A. Courville, Y. Bengio, and P. Vincent. Why does unsupervised pre-training help deep learning? In Y. W. Teh and M. Titterington, editors, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR W&CP*, pages 201–208, 2010.

A. Fischer and C. Igel. Contrastive divergence learning may diverge when training restricted Boltzmann machines. *Frontiers in Computational Neuroscience. Bernstein Conference on Computational Neuroscience (BCCN)*, 2009.

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, 2010a.

A. Fischer and C. Igel. Challenges in training restricted Boltzmann machines. In B. Hammer and T. Villmann, editors, *New Challenges in Neural Computation (NC^2)*, number 04/2010 in Machine Learning Reports, pages 11–24. 2010b.

A. Fischer and C. Igel. Bounding the bias of contrastive divergence learning. *Neural Computation*, 23:664–673, 2011a.

A. Fischer and C. Igel. Parallel tempering, importance sampling, and restricted Boltzmann machines. In *5th Workshop on Theory of Randomized Search Heuristics (ThRaSH)*, 2011b. Online abstract.

A. Fischer and C. Igel. Training restricted Boltzmann machines: An introduction. *Pattern Recognition*, 47:25–39, 2014.

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.

A. Gelman and X.-L. Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185, 1998.

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.

C. J. Geyer. Markov chain Monte Carlo maximum likelihood. In E. Kerami, editor, *Proceedings of the 23rd Symposium on the Interface of Computing Science and Statistics*, pages 156–163. Interface Foundation of North America, 1991.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434, 2007a.

G. E. Hinton. Boltzmann machine. *Scholarpedia*, 2(5):1668, 2007b.

G. E. Hinton. A practical guide to training restricted Boltzmann machines. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 599–619. Springer, 2012.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

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.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

J. Kivinen and C. Williams. Multiple texture Boltzmann machines. In N. Lawrence and M. Girolami, editors, *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 22 of *JMLR W&CP*, pages 638–646, 2012.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

O. Krause, A. Fischer, T. Glasmachers, and C. Igel. Approximation properties of DBNs with binary hidden units and real-valued visible units. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *JMLR W&CP*, pages 419–426, 2013.

H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Proceedings of the 25th International Conference on Machine learning (ICML)*, pages 536–543. ACM, 2008.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 473–480. ACM, 2007.

H. Larochelle, M. I. Mandel, R. Pascanu, and Y. Bengio. Learning algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, 13:643–669, 2012.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

N. Le Roux and Y. Bengio. Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

N. Le Roux and Y. Bengio. Deep belief networks are compact universal approximators. *Neural Computation*, 22(8):2192–2207, 2010.

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.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a.

Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller. Efficient backprop. *Neural Networks: Tricks of the trade*, 1998b.

H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 609–616. ACM, 2009a.

H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems (NIPS 22)*, pages 1096–1104. MIT Press, 2009b.

J. Q. Li and A. R. Barron. Mixture density estimation. In S. A. Solla, T. K. Leen, and K. R. Müller, editors, *Advances in Neural Information Processing Systems (NIPS 12)*, pages 279–285. MIT Press, 2000.

M. N. Lingenheil, R. Denschlag, G. Mathias, and P. Tavan. Efficiency of exchange schemes in replica exchange. *Chemical Physics Letters*, 478:80 – 84, 2009.

J. S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6:113–119, 1996.

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.

D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

N. Madras and D. Randall. Markov chain decomposition for convergence rate analysis. *The Annals of Applied Probability*, 12(581–606), 2002.

N. Madras and Z. Zheng. On the swapping algorithm. *Random Structures Algorithms*, 22:66–97, 2003.

X.-L. Meng and W. H. Wong. Simulating ratios of normalizing constants via a simple identity: A theoretical explanation. *Statistica Sinica*, 6:831–860, 1996.

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.

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.

G. Montavon and K. Müller. Deep Boltzmann machines and the centering trick. *Lecture Notes in Computer Science (LNCS)*, 7700:621–637, 2012.

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.

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.

R. M. Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

R. M. Neal. Estimating ratios of normalizing constants using linked importance sampling. *ArXiv Mathematics e-prints*, 2005.

Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. Technical report, CoRR, abs/1106.3708v2, 2013.

P. H. Peskun. Optimum Monte-Carlo sampling using Markov chains. *Biometrika*, 60 (3):607–612, 1973.

T. Raiko, H. Valpola, and Y. LeCun. Deep learning made easier by linear transformations in perceptrons. *Journal of Machine Learning Research*, 22:924–332, 2012.

M. A. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, 2007. IEEE.

M. Riedmiller. Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16(5):265–278, 1994.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986a.

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, 1986b.

R. Salakhutdinov. Learning and evaluating Boltzmann machine. Technical report, University of Toronto, 2008.

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. MIT Press, 2009.

R. Salakhutdinov and G. E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In M. Meila and X. Shen, editors, *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 2 of *JMLR W&CP*, pages 412 – 419, 2007.

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. MIT Press, 2009a.

R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In D. van Dyk and M. Welling, editors, *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5 of *JMLR W&CP*, pages 448–455, 2009b.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Proceedings of the International Conference on Machine Learning (ICML)*, volume 25. ACM, 2008.

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.

J. Schlüter and C. Osendorfer. Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine. In *Proceedings of the 10th International Conference on Machine Learning and Applications (ICMLA)*, 2011.

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. MIT Press, 2009.

H. Schulz, A. Müller, and S. Behnke. Investigating convergence of restricted Boltzmann machine learning. *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

B. Schwehn. Using the natural gradient for training restricted Boltzmann machines. Master's thesis, University of Edinburgh, Edinburgh, 2010.

N. R. Shirts, E. Bair, G. Hooker, and V. S. Pande. Equilibrium free energies from nonequilibrium measurements using maximum-likelihood methods. *Physical Review Letters*, 91:140601, 2003.

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.

S. Sukhbaatar, T. Makino, K. Aihara, and T. Chikayama. Robust generation of dynamical patterns in human motion by a deep belief nets. In C. S. Ong and T. B. Ho, editors, *Proceedings of the 3rd Asian Conference on Machine Learning (ACML)*, JMLR W&CP, pages 231–246, 2011.

I. Sutskever and T. Tieleman. On the convergence properties of contrastive divergence. In Y. W. Teh and M. Titterington, editors, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR W&CP*, pages 789–795, 2010.

R. H. Swendsen and J.-S. Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57:2607–2609, 1986.

R. H. Swendsen and J.-S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58:86–88, 1987.

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.

Y. Tang and I. Sutskever. Data normalization in the learning of restricted Boltzmann machines. Technical report, Department of Computer Science, University of Toronto, 2011.

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.

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 International Conference on Machine Learning (ICML)*, pages 1025–1032. ACM, 2009.

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.

G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *Computer Vision – ECCV 2010*, volume 6316 of *LNCS*, pages 140–153. Springer, 2010.

M. B. Thompson. A comparison of methods for computing autocorrelation time. Technical Report 1007, Department of Statistics, University of Toronto, 2010.

M. B. Thompson. Introduction to SamplerCompare. *Journal of Statistical Software*, 43(12), 2011.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Proceedings of the 25th International Conference on Machine learning (ICML)*, pages 1064–1071. ACM, 2008.

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, *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 1033–1040. ACM, 2009.

S. N. Tran, D. Wolff, T. Weyd, and A. Garcez. Feature preprocessing with RBMs for music similarity learning. In *Proceedings of the AES 53rd International Conference on Semantic Audio*, 2014.

J.-S. Wang and R. H. Swendsen. Cluster Monte Carlo algorithms. *Physica A: Statistical Mechanics and its Applications*, 167(3):565 – 579, 1990.

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.

M. Welling. Product of experts. *Scholarpedia*, 2(10):3879, 2007.

M. Welling, M. Rosen-Zvi, and G. E. 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.

U. Wolff. Collective Monte Carlo updating for spin systems. *Physical Review Letters*, 62:361–364, 1989.

D. Woodard, S. Schmidler, and M. Huber. Sufficient conditions for torpid mixing of parallel and simulated tempering. *Electronic Journal of Probability*, 14:780–804, 2009a.

D. B. Woodard, S. C. Schmidler, and M. Huber. Conditions for rapid mixing of parallel and simulated tempering on multimodal distributions. *The Annals of Applied Probability*, 19:617–640, 2009b.

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.

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.

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.

A. J. Zeevi and R. Meir. Density estimation through convex combinations of densities: Approximation and estimation bounds. *Neural Networks*, 10(1):99 – 109, 1997.

List of Publications

Oswin Krause, Asja Fischer, and Christian Igel. On Bennett's acceptance ratio for estimating the partition function of restricted Boltzmann machines, *submitted*

Jan Melchior, Asja Fischer, and Laurenz Wiskott. How to center restricted Boltzmann machines, *submitted*

Asja Fischer and Christian Igel. A bound for the convergence rate of parallel tempering for sampling restricted Boltzmann machines, *submitted*.

Asja Fischer and Christian Igel. Training restricted Boltzmann machines: An introduction. *Pattern Recognition* 47, pp. 25-39, 2014

Kai Brügge, Asja Fischer, and Christian Igel. The flip-the-state transition operator for restricted Boltzmann machines. *Machine Learning* 13, pp. 53-69, 2013

Oswin Krause, Asja Fischer, Tobias Glasmachers, and Christian Igel. Approximation properties of DBNs with binary hidden units and real-valued visible units. In S. Dasgupta and D. McAllester, eds.: *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, JMLR W&CP, 28(1), pp. 419-426, 2013

Asja Fischer and Christian Igel. An introduction to restricted Boltzmann machines. In Luis Alvarez, Marta Mejail, Luis Gomez, and Julio Jacobo, eds.: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP 2012)*, LNCS 7441, pp. 14-36, Springer-Verlag, 2012

Asja Fischer and Christian Igel. Bounding the bias of contrastive divergence learning. *Neural Computation* 23, pp. 664-673, 2011

Asja Fischer and Christian Igel. Training RBMs based on the signs of the CD approximation of the log-likelihood derivatives. In Michel Verleysen, ed.: *19th European Symposium on Artificial Neural Networks (ESANN 2011)*, pp. 495-500, Belgium: d-side publications, 2011

Asja Fischer and Christian Igel. Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis, eds.: *International Conference on Artificial Neural Networks (ICANN 2010)*, LNCS 6354, pp. 208-217, Springer-Verlag, 2010

Asja Fischer and Christian Igel. Challenges in training restricted Boltzmann machines. In Barbara Hammer and Thomas Villmann, eds.: *New Challenges in Neural Computation (NC2)*, Machine Learning Reports 04/2010, pp. 1124, 2010

Asja Fischer and Christian Igel. Contrastive divergence learning may diverge when training restricted Boltzmann machines. Frontiers in Computational Neuroscience. Conference Abstract: *Bernstein Conference on Computational Neuroscience (BCCN 2009)*, 2009