Journal of Machine Learning Research 17 (2016) 1-61 Submitted 6/14; Revised 7/15; Published 6/16

How to Center Deep Boltzmann Machines

Jan Melchior Jan.Melchior@ini.rub.de
Asja Fischer Asja.Fischer@ini.rub.de
Laurenz Wiskott Laurenz.Wiskott@ini.rub.de
Institut für Neuroinformatik
Ruhr-Universität Bochum
Bochum, D-44801, Germany

Editor: Yoshua Bengio

Abstract

This work analyzes centered Restricted Boltzmann Machines (RBMs) and centered Deep Boltzmann Machines (DBMs), where centering is done by subtracting offset values from visible and hidden variables. We show analytically that (i) centered and normal Boltzmann Machines (BMs) and thus RBMs and DBMs are different parameterizations of the same model class, such that any normal BM/RBM/DBM can be transformed to an equivalent centered BM/RBM/DBM and *vice versa*, and that this equivalence generalizes to artificial neural networks in general, (ii) the expected performance of centered binary BMs/RBMs/DBMs is invariant under simultaneous flip of data and offsets, for any offset value in the range of zero to one, (iii) centering can be reformulated as a different update rule for normal BMs/RBMs/DBMs, and (iv) using the enhanced gradient is equivalent to setting the offset values to the average over model and data mean. Furthermore, we present numerical simulations suggesting that (i) optimal generative performance is achieved by subtracting mean values from visible as well as hidden variables, (ii) centered binary RBMs/DBMs reach significantly higher log-likelihood values than normal binary RBMs/DBMs, (iii) centering variants whose offsets depend on the model mean, like the enhanced gradient, suffer from severe divergence problems, (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 severe divergence, (v) on a similar level of log-likelihood values centered binary RBMs/DBMs have smaller weights and bigger bias parameters than normal binary RBMs/DBMs, (vi) centering leads to an update direction that is closer to the natural gradient, which is extremely efficient for training as we show for small binary RBMs, (vii) centering eliminates the need for greedy layer-wise pre-training of DBMs, which often even deteriorates the results independently of whether centering is used or not, and (ix) centering is also beneficial for auto encoders.

Keywords: centering, restricted Boltzmann machine, deep Boltzmann machine, generative model, artificial neural network, auto encoder, enhanced gradient, natural gradient, stochastic maximum likelihood, contrastive divergence, parallel tempering

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 with only 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, 2010; Schulz et al., 2010; Fischer and Igel, 2011). To overcome the divergence problem Desjardins et al. (2010) and Cho et al. (2010) have proposed to use parallel tempering (Swendsen and Wang, 1986), 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 this 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 known as the 'centering trick', which was originally proposed for feed forward neural networks (LeCun et al., 1998; Schraudolph, 1998). It has recently also been applied to the visible and hidden variables of Deep Boltzmann Machines (DBM) (Montavon and Müller, 2012) where it has been shown to lead to a better conditioned optimization problem. Furthermore, the learned features 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 has been proposed by Raiko et al. (2012). The authors could show that the gradient under this transformation gets 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). Since the exact natural gradient is intractable already for rather small RBMs, Schwehn (2010) and Ollivier et al. (2011) have trained binary RBMs and Desjardins et al. (2013) binary DBMs using approximations of the natural gradient obtained by Markov chain Monte Carlo methods. However, due to

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

the computational overhead the practical relevance of the natural gradient for RBM/DBM training remains questionable. Another approach related to the centering trick is batch normalization, which has recently been proposed by Ioffe and Szegedy (2015) and aims at removing first and second order statistics of the pre synaptic activation in a feed forward network.

In this article[2] we give a unified view on centering, revealing that the methods proposed by Cho et al. (2011), Tang and Sutskever (2011), and Montavon and Müller (2012) can all be considered as different ways of applying the centering trick to RBMs and DBMs. Furthermore, we analyze the properties and performance of different centering variants. In Section 2, 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 3, we discuss the theoretical properties of centered RBMs, show that centering can be reformulated as a different update rule for normal RBMs, that the enhanced gradient is a particular form of centering, and finally that centering and its properties naturally extend to DBMs and BMs. Furthermore, in Section 4, we show that centering is an alternative parameterization for arbitrary Artificial Neural Networks (ANNs) in general and we discuss how the parameters of centered and normal ANNs should be initialized. Our experimental setups are described in Section 5 before we empirically analyze the performance of centered RBMs with different initializations, offset parameters, sampling methods, and learning rates in Section 6. This empirical analysis includes a comparison of the centered gradient with the natural gradient and extensive experiments on 10 real world data sets. In addition we have also performed experiments with DBMs and Auto encoders (AEs) on the 10 real work data sets. Finally our work is concluded in Section 7.

2. Restricted Boltzmann Machines

An RBM (Smolensky, 1986) is a bipartite undirected graphical model with a set of N visible and M hidden variables taking values $\mathbf{x} = (x_1, ..., x_N)$ and $\mathbf{h} = (h_1, ..., h_M)$, respectively. Since an RBM is a Markov random field, its joint probability distribution is given by a Gibbs distribution

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

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

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

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

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

2. Previous versions of this work have been published as an eprint (Melchior et al., 2013) and as part of the PhD thesis by Fischer (2014).

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

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

where $\langle\cdot\rangle_m$ is the expectation under $p(\mathbf{x},\mathbf{h})$ and $\langle\cdot\rangle_d$ is the expectation under $p(\mathbf{h}|\mathbf{x})p_e(\mathbf{x})$ 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

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

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 Contrastive Divergence (CD) (Hinton et al., 2006). 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, that is only used for Gibbs sampling during learning. The advanced sampling method Parallel Tempering (PT) (Swendsen and Wang, 1986) introduces additional 'tempered' Gibbs chains corresponding to smoothed versions of $p(\mathbf{x},\mathbf{h})$. 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, 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 with temperature $T = 1$ (a first theoretical analysis of the mixing rate of PT for sampling in RBMs has been given by Fischer and Igel, 2015). 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., 2010; Cho et al., 2010) 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.

2.1 Enhanced Gradient

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

When transforming the state $(\mathbf{x},\mathbf{h})$ of a binary RBM by flipping some of its variables (that is $\tilde{x}_i = 1 - x_i$, and $\tilde{h}_j = 1 - h_j$ for some i, j), yielding a new state $(\tilde{\mathbf{x}},\tilde{\mathbf{h}})$, one can transform the parameters $\boldsymbol{\theta}$ of the RBM to $\tilde{\boldsymbol{\theta}}$ such that $E(\mathbf{x},\mathbf{h}|\boldsymbol{\theta}) = E(\tilde{\mathbf{x}},\tilde{\mathbf{h}}|\tilde{\boldsymbol{\theta}}) + \mathit{const}$ and thus $p(\mathbf{x},\mathbf{h}|\boldsymbol{\theta}) = p(\tilde{\mathbf{x}},\tilde{\mathbf{h}}|\tilde{\boldsymbol{\theta}})$. 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 differs 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 $(\mathbf{x},\mathbf{h})$. Cho et al. (2011) have proposed the enhanced gradient as a weighted sum of these 2^{N+M} parameter updates, which for their choice of weighting is given by

$$\nabla_e\mathbf{W} = \langle(\mathbf{x}-\langle\mathbf{x}\rangle_d)(\mathbf{h}-\langle\mathbf{h}\rangle_d)^T\rangle_d - \langle(\mathbf{x}-\langle\mathbf{x}\rangle_m)(\mathbf{h}-\langle\mathbf{h}\rangle_m)^T\rangle_m \ , \tag{1}$$

$$\nabla_e\mathbf{b} = \langle\mathbf{x}\rangle_d - \langle\mathbf{x}\rangle_m - \nabla_e\mathbf{W}\frac{1}{2}\left(\langle\mathbf{h}\rangle_d + \langle\mathbf{h}\rangle_m\right) \ , \tag{2}$$

$$\nabla_e\mathbf{c} = \langle\mathbf{h}\rangle_d - \langle\mathbf{h}\rangle_m - \nabla_e\mathbf{W}^T\frac{1}{2}\left(\langle\mathbf{x}\rangle_d + \langle\mathbf{x}\rangle_m\right) \ . \tag{3}$$

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 have reported more stable training under various settings in terms of the LL estimate and classification accuracy.

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 $\{p(\mathbf{x}|\boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}$, 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 $||\boldsymbol{\theta}||_{\mathcal{I}(\boldsymbol{\theta})} = \sqrt{\sum \theta_k \mathcal{I}_{kl}\left(\boldsymbol{\theta}\right)\theta_l}$, 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 $p(\mathbf{x}|\boldsymbol{\theta})$ is given by

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

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

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

The inverse Fisher information matrix corrects the direction and length of the standard gradient to achieve the largest change of the objective function (here the LL) for an infinitesimal small step $\delta\boldsymbol{\theta}$ from $p(\mathbf{x}|\boldsymbol{\theta})$ to $p(\mathbf{x}|\boldsymbol{\theta}+\boldsymbol{\delta\theta})$ in terms of the Kullback-Leibler divergence (Amari, 1998). This makes the natural gradient independent of the parameterization leading to an invariance to arbitrary coordinate transformations such as flipping variables, which makes it 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., 2011) are given by

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

where $Cov_m(\cdot,\cdot)$ denotes the covariance under the model distribution. 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., 2011; Desjardins et al., 2013). Furthermore, a diagonal approximation of the Fisher information matrix could be used. However, the approximation of the natural gradient for RBMs and DBMs is still computationally very expensive so that the practical usability remains questionable (Schwehn, 2010; Ollivier et al., 2011; Desjardins et al., 2013).

3. Centered Restricted Boltzmann Machines

Inspired by the centering trick proposed by LeCun et al. (1998), Tang and Sutskever (2011) have addressed the problem that RBMs perform differently on the *MNIST* and *1-MNIST* data set 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 $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_N)$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$, respectively, can be formulated as

$$E(\mathbf{x}, \mathbf{h}) = -(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{b} - \mathbf{c}^T (\mathbf{h} - \boldsymbol{\lambda}) - (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{W} (\mathbf{h} - \boldsymbol{\lambda}) \ . \tag{4}$$

By setting both offsets to zero one retains the normal binary RBM. Setting $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_d$ and $\boldsymbol{\lambda} = \mathbf{0}$ leads to the model introduced by Tang and Sutskever (2011), and by setting $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_d$ 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

$$p(x_i = 1|\mathbf{h}) = \sigma(\mathbf{w_{i*}}(\mathbf{h} - \boldsymbol{\lambda}) + b_i) \ , \tag{5}$$

$$p(h_j = 1|\mathbf{x}) = \sigma((\mathbf{x} - \boldsymbol{\mu})^T \mathbf{w_{*j}} + c_j) \ , \tag{6}$$

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

$$\nabla \mathbf{W} = \langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_d - \langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_m \ , \tag{7}$$

$$\nabla \mathbf{b} = \langle \mathbf{x} - \boldsymbol{\mu} \rangle_d - \langle \mathbf{x} - \boldsymbol{\mu} \rangle_m = \langle \mathbf{x} \rangle_d - \langle \mathbf{x} \rangle_m \ , \tag{8}$$

$$\nabla \mathbf{c} = \langle \mathbf{h} - \boldsymbol{\lambda} \rangle_d - \langle \mathbf{h} - \boldsymbol{\lambda} \rangle_m = \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m \ . \tag{9}$$

$\nabla\mathbf{b}$ 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 Appendix A) that the gradient of a centered BM is invariant to flip transformations if a flip of x_i to $1-x_i$ implies a change of μ_i to $1-\mu_i$, and in the case of RBMs a flip h_j to $1-h_j$ implies a change of λ_j to $1-\lambda_j$. This holds for $\mu_i = 0.5$, $\lambda_j = 0.5$ but remarkably also for the expectation values over x_i and h_j under any distribution. Moreover, if the offsets are set to the expectation values, centered RBMs get also invariant to shifts

Algorithm 1: Training centered RBMs

```
1  Initialize $\mathbf{W}$ ;                                   /* i.e., $\mathbf{W} \leftarrow \mathcal{N}(0, 0.01)^{N\times M}$ */
2  Initialize $\boldsymbol{\mu}, \boldsymbol{\lambda}$ ;       /* i.e., $\boldsymbol{\mu} \leftarrow \langle\texttt{data}\rangle, \boldsymbol{\lambda} \leftarrow \mathbf{0.5}$ */
3  Initialize $\mathbf{b}, \mathbf{c}$ ;                       /* i.e., $\mathbf{b} \leftarrow \sigma^{-1}(\boldsymbol{\mu}), \mathbf{c} \leftarrow \sigma^{-1}(\boldsymbol{\lambda})$ */
4  Initialize $\eta, \nu_\mu, \nu_\lambda$ ;                    /* i. e., $\eta, \nu_\mu, \nu_\lambda \in \{0.001, ..., 0.1\}$ */
5  repeat
6      foreach batch in data do
7          foreach sample $\mathbf{x}_d$ in batch do
8              Calculate $\mathbf{h}_d = p(\mathbf{h} = \mathbf{1}|\mathbf{x}_d)$ ;          /* ▷ Eq. (6) */
9              Sample $\mathbf{x}_m$ from RBM ;                          /* i.e., PCD-1, ▷ Eqs. (5), (6) */
10             Calculate $\mathbf{h}_m = p(\mathbf{h} = \mathbf{1}|\mathbf{x}_m)$ ;          /* ▷ Eq. (6) */
11             Store $\mathbf{x}_m, \mathbf{h}_d, \mathbf{h}_m$
12         Estimate $\boldsymbol{\mu}_{batch}$ ;                         /* i.e. $\boldsymbol{\mu}_{batch} \leftarrow \langle\mathbf{x}_d\rangle$ */
13         Estimate $\boldsymbol{\lambda}_{batch}$ ;                     /* i.e. $\boldsymbol{\lambda}_{batch} \leftarrow \langle\mathbf{h}_d\rangle$ */
           /* Transform parameters with respect to the new offsets */
14         $\mathbf{b} \leftarrow \mathbf{b} + \nu_\lambda \mathbf{W}(\boldsymbol{\lambda}_{batch} - \boldsymbol{\lambda})$ ;   /* ▷ Eq. (11), $\tilde{\boldsymbol{\lambda}} = (1-\nu_\lambda)\boldsymbol{\lambda} + \nu_\lambda\boldsymbol{\lambda}_{batch}$ */
15         $\mathbf{c} \leftarrow \mathbf{c} + \nu_\mu \mathbf{W}^T(\boldsymbol{\mu}_{batch} - \boldsymbol{\mu})$ ;   /* ▷ Eq. (12), $\tilde{\boldsymbol{\mu}} = (1-\nu_\mu)\boldsymbol{\mu} + \nu_\mu\boldsymbol{\mu}_{batch}$ */
           /* Update offsets using exp. moving average with sliding factors
              $\nu_\mu$ and $\nu_\lambda$ */
16         $\boldsymbol{\mu} \leftarrow (1-\nu_\mu)\boldsymbol{\mu} + \nu_\mu\boldsymbol{\mu}_{batch}$
17         $\boldsymbol{\lambda} \leftarrow (1-\nu_\lambda)\boldsymbol{\lambda} + \nu_\lambda\boldsymbol{\lambda}_{batch}$
           /* Update parameters using the gradient with learning rate $\eta$ */
18         $\nabla\mathbf{W} \leftarrow \langle(\mathbf{x}_d - \boldsymbol{\mu})(\mathbf{h}_d - \boldsymbol{\lambda})^T\rangle - \langle(\mathbf{x}_m - \boldsymbol{\mu})(\mathbf{h}_m - \boldsymbol{\lambda})^T\rangle$ ;   /* ▷ Eq. (7) */
19         $\nabla\mathbf{b} \leftarrow \langle\mathbf{x}_d\rangle - \langle\mathbf{x}_m\rangle$ ;          /* ▷ Eq. (8) */
20         $\nabla\mathbf{c} \leftarrow \langle\mathbf{h}_d\rangle - \langle\mathbf{h}_m\rangle$ ;          /* ▷ Eq. (9) */
21         $\mathbf{W} \leftarrow \mathbf{W} + \eta\nabla\mathbf{W}$
22         $\mathbf{b} \leftarrow \mathbf{b} + \eta\nabla\mathbf{b}$
23         $\mathbf{c} \leftarrow \mathbf{c} + \eta\nabla\mathbf{c}$
24 until stopping criteria is met;
   /* Transform network to a normal binary RBM if desired */
25 $\mathbf{b} \leftarrow \mathbf{b} - \mathbf{W}\boldsymbol{\lambda}$ ;          /* ▷ Eq. (11) */
26 $\mathbf{c} \leftarrow \mathbf{c} - \mathbf{W}^T\boldsymbol{\mu}$ ;          /* ▷ Eq. (12) */
27 $\boldsymbol{\mu} \leftarrow \mathbf{0}$
28 $\boldsymbol{\lambda} \leftarrow \mathbf{0}$
```

of variables (see Section 4). Note that these properties of centered RBMs naturally extend to centered BMs and DBMs (see Section 3.2, Appendix A and B).

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 RBMs needs to update the offset values to match the expectations under the distribution that has changed through 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 $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\lambda}}$ by

$$\tilde{\mathbf{W}} = \mathbf{W} \ , \tag{10}$$

$$\tilde{\mathbf{b}} = \mathbf{b} + \mathbf{W}\left(\tilde{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\right) \ , \tag{11}$$

$$\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{W}^T\left(\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}\right) \ , \tag{12}$$

such that $E(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = E(\mathbf{x}, \mathbf{h}|\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\lambda}}) + const$, is guaranteed (see Appendix B for a derivation for a more general BM). Obviously, this can also be used to transform a centered RBM to a normal RBM and *vice versa*, highlighting that centered and normal RBMs are just different parameterizations of the same model class.

If the intractable model mean is used for an offset, it has 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 $\langle \mathbf{x} \rangle_m$ one could approximate the mean values either 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.[3]

Algorithm 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 $\langle \mathbf{x}_d \rangle$ for the average value of data samples $\mathbf{x}_d$ in the current batch, which is used as an approximation for the expectation of $\mathbf{x}$ under the data distribution, that is $\langle \mathbf{x} \rangle_d$. Similarly, $\langle \mathbf{h}_d \rangle = \langle p\left(\mathbf{h} = \mathbf{1}|\mathbf{x}_d\right) \rangle$ approximates $\langle \mathbf{h} \rangle_d$, where $p\left(\mathbf{x} = \mathbf{1}|\mathbf{h}\right)$ denotes the vector containing the elements $p\left(x_i = 1|\mathbf{h}\right)$. Note that in Algorithm 1 the update of the offsets is performed before the gradient is calculated, such that gradient and reparameterization both use the current samples. 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. Thus, while the gradient still uses the current samples the reparameterization is 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 1 an exponentially moving average for the approximation of $\boldsymbol{\mu}$ is obtained if the corresponding sliding factor ν_μ is set to $0 < \nu_\mu < 1$ or prevented if $\nu_\mu = 1$ and equivalently for $\boldsymbol{\lambda}$ and ν_λ. The effects of using an exponentially moving average are empirically analyzed in Section 6.5.

3. This can be proven analogously to the proof of proposition 1 in the work of Swersky et al. (2010).

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 RBMs. Similar to the derivation of the enhanced gradient we can transform a normal RBM to a centered RBM, perform a gradient update, and transform the RBM back (see Appendix B for a derivation for the more general BM). This yields the following parameter updates, which we refer to as centered gradient

$$\nabla_c \mathbf{W} = \langle (\mathbf{x}-\boldsymbol{\mu})(\mathbf{h}-\boldsymbol{\lambda})^T \rangle_d - \langle (\mathbf{x}-\boldsymbol{\mu})(\mathbf{h}-\boldsymbol{\lambda})^T \rangle_m \ , \tag{13}$$

$$\nabla_c \mathbf{b} = \langle \mathbf{x} \rangle_d - \langle \mathbf{x} \rangle_m - \nabla_c \mathbf{W} \boldsymbol{\lambda} \ , \tag{14}$$

$$\nabla_c \mathbf{c} = \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m - \nabla_c \mathbf{W}^T \boldsymbol{\mu} \ . \tag{15}$$

Note that by setting $\boldsymbol{\mu} = \frac{1}{2}(\langle \mathbf{x} \rangle_d + \langle \mathbf{x} \rangle_m)$ and $\boldsymbol{\lambda} = \frac{1}{2}(\langle \mathbf{h} \rangle_d + \langle \mathbf{h} \rangle_m)$ the centered gradient becomes equal to the enhanced gradient (see Appendix C). 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 in (Cho et al., 2013a).

Algorithm 2: Training RBMs using the centered gradient

```
1  Initialize $\mathbf{W}$ ;                                  /* i.e. $\mathbf{W} \leftarrow \mathcal{N}(0, 0.01)^{N \times M}$ */
2  Initialize $\boldsymbol{\mu}, \boldsymbol{\lambda}$ ;      /* i.e. $\boldsymbol{\mu} \leftarrow \langle \mathtt{data} \rangle, \boldsymbol{\lambda} \leftarrow \mathbf{0.5}$ */
3  Initialize $\mathbf{b}, \mathbf{c}$ ;                     /* i.e. $\mathbf{b} \leftarrow \sigma^{-1}(\boldsymbol{\mu}), \mathbf{c} \leftarrow \sigma^{-1}(\boldsymbol{\lambda})$ */
4  Initialize $\eta, \nu_\mu, \nu_\lambda$ ;                 /* i.e. $\eta, \nu_\mu, \nu_\lambda \in \{0.001, ..., 0.1\}$ */
5  repeat
6      foreach batch in data do
7          foreach sample $\mathbf{x}_d$ in batch do
8              Calculate $\mathbf{h}_d = p(\mathbf{h} = \mathbf{1} | \mathbf{x}_d)$ ;      /* ▷ Eq. (6) */
9              Sample $\mathbf{x}_m$ from RBM ;             /* i.e., PCD-1, ▷ Eqs. (5), (6) */
10             Calculate $\mathbf{h}_m = p(\mathbf{h} = \mathbf{1} | \mathbf{x}_m)$ ;      /* ▷ Eq. (6) */
11             Store $\mathbf{x}_m, \mathbf{h}_d, \mathbf{h}_m$
12         Estimate $\boldsymbol{\mu}_{batch}$ ;            /* i.e. $\boldsymbol{\mu}_{batch} \leftarrow \langle \mathbf{x}_d \rangle$ */
13         Estimate $\boldsymbol{\lambda}_{batch}$ ;        /* i.e. $\boldsymbol{\lambda}_{batch} \leftarrow \langle \mathbf{h}_d \rangle$ */
           /* Update offsets using exp. moving averages with sliding factors
              $\nu_\mu$ and $\nu_\lambda$ */
14         $\boldsymbol{\mu} \leftarrow (1 - \nu_\mu)\boldsymbol{\mu} + \nu_\mu \boldsymbol{\mu}_{batch}$
15         $\boldsymbol{\lambda} \leftarrow (1 - \nu_\lambda)\boldsymbol{\lambda} + \nu_\lambda \boldsymbol{\lambda}_{batch}$
           /* Update parameters using the centered gradient with learning
              rate $\eta$ */
16         $\nabla_c \mathbf{W} \leftarrow \langle (\mathbf{x}_d - \boldsymbol{\mu})(\mathbf{h}_d - \boldsymbol{\lambda})^T \rangle - \langle (\mathbf{x}_m - \boldsymbol{\mu})(\mathbf{h}_m - \boldsymbol{\lambda})^T \rangle$ ;   /* ▷ Eq. (13) */
17         $\nabla_c \mathbf{b} \leftarrow \langle \mathbf{x}_d \rangle - \langle \mathbf{x}_m \rangle - \nabla_c \mathbf{W} \boldsymbol{\lambda}$ ;   /* ▷ Eq. (14) */
18         $\nabla_c \mathbf{c} \leftarrow \langle \mathbf{h}_d \rangle - \langle \mathbf{h}_m \rangle - \nabla_c \mathbf{W}^T \boldsymbol{\mu}$ ;   /* ▷ Eq. (15) */
19         $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla_c \mathbf{W}$
20         $\mathbf{b} \leftarrow \mathbf{b} + \eta \nabla_c \mathbf{b}$
21         $\mathbf{c} \leftarrow \mathbf{c} + \eta \nabla_c \mathbf{c}$
22 until stopping criteria is met;
```

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 $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_m$ or $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_m$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_d$. However, these offsets result in different update rules for the bias parameters.

Algorithm 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 1. Both algorithms can easily be extended to RBMs with other types of units, DBMs and BMs.

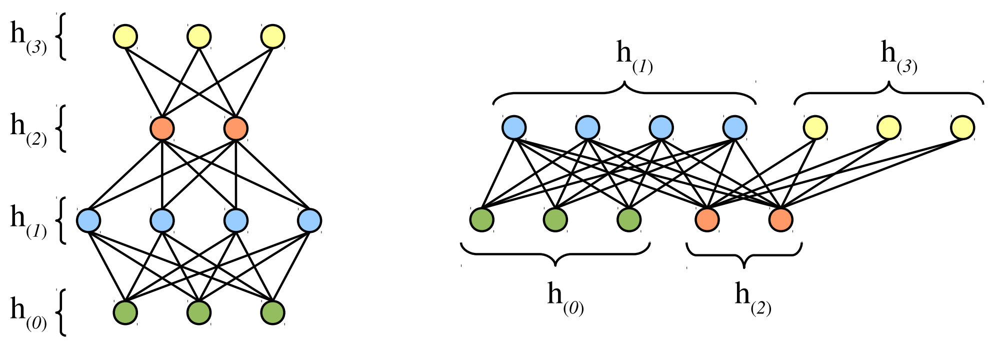
3.2 Centered Deep Boltzmann Machines

A DBM (Salakhutdinov and Hinton, 2009) is a deep undirected graphical model with several hidden layers where successive layers have a bipartite connectivity structure. Therefore, a DBM can be seen as a jointly trained stack of several RBMs and thus as a natural extension of RBMs. A centered binary DBM with L layers $\mathbf{h}_{(0)}, \cdots, \mathbf{h}_{(L)}$ (where $\mathbf{h}_{(0)}$ corresponds to the visible layer) represents a Gibbs distribution with energy

$$E\left(\mathbf{h}_{(0)}, \cdots, \mathbf{h}_{(L)}\right) = -\sum_{l=0}^{L} \left(\mathbf{h}_{(l)} - \boldsymbol{\lambda}_{(l)}\right)^T \mathbf{b}_{(l)} - \sum_{l=0}^{L-1} \left(\mathbf{h}_{(l)} - \boldsymbol{\lambda}_{(l)}\right)^T \mathbf{W}_{(l)} \left(\mathbf{h}_{(l+1)} - \boldsymbol{\lambda}_{(l+1)}\right),$$

where each layer l has a bias $\mathbf{b}_{(l)}$, an offset $\boldsymbol{\lambda}_{(l)}$ and is connected to layer $l+1$ by weight matrix $\mathbf{W}_l$.

The derivations, proofs and algorithms given in this work for RBMs/BMs automatically extend to DBMs since each DBM can be transformed to an RBM with restricted connections and partially unknown input data. This is illustrated for a DBM with four layers in Figure 1. As a consequence of this relation DBMs can essentially be trained in the same way as RBMs but also suffer from the same problems as described before. The major difference in DBM



(a) Deep network version. (b) Shallow network version.

Figure 1: Example for (a) a DBM or an ANN with four layers $h_{(0)}, \cdots, h_{(3)}$ and (b) the equivalent two layer shallow version of the same network with restricted connections and unknown input $h_{(2)}$.

training compared to RBM training is that the expectation under the data distribution in the LL gradient of DBMs gets intractable. Due to the partially missing input data the factorization trick used for the calculation of expectation under the data distribution in RBMs cannot be applied anymore. Instead the term is approximated by running a mean field estimation until convergence (Salakhutdinov and Hinton, 2009), which corresponds to approximating the gradient of the variational lower bound of the LL. Furthermore, it is common to pre-train DBMs in a greedy layer wise fashion using RBMs (Salakhutdinov and Hinton, 2009; Hinton and Salakhutdinov, 2012).

4. Centering in Artificial Neural Networks in General

Removing the mean from visible and hidden units was originally proposed for feed forward neural networks (LeCun et al., 1998; Schraudolph, 1998). When this idea was applied to the visible units of RBMs (Tang and Sutskever, 2011) the model was reparameterized such that the probability distribution defined by the normal and centered RBM stayed the same.

In this section we generalize this concept to show that centering is an alternative parameterization for arbitrary ANN architectures in general, if the network is reparameterized accordingly. Consider the centered artificial neuron model

$$o_j = \phi_j\left(\sum_i w_{ij}\left(a_i - \mu_i\right) + c_j\right), \tag{16}$$

where the output o_j of the jth neuron depends on its activation function ϕ_j, bias term c_j and weights w_{ij} with associated inputs a_i and their corresponding offsets μ_i. Such neurons can be used to construct arbitrary network architectures using undirected, directed and recurrent connections, which can then be optimized with respect to a chosen loss.

Two different ANNs that represent exactly the same functional input-output mapping can be considered as different parameterizations of the same model. Thus, a centered ANN is just a different parameterization of an uncentered ANN if we can show that their functional input-output mappings are the same. This can be guaranteed in general if all corresponding units in a centered and an uncentered ANN have the same mapping from inputs to outputs. If the offset μ_i is changed to $\tilde{\mu}_i = \mu_i + \nabla\mu_i$ then the output of the centered artificial neuron (16) becomes

$$\begin{aligned}\phi_j\left(\sum_i w_{ij}\left(a_i - \tilde{\mu}_i\right) + c_j\right) &= \phi_j\left(\sum_i w_{ij}\left(a_i - \left(\mu_i + \nabla\mu_i\right)\right) + c_j\right) \\ &= \phi_j\left(\sum_i w_{ij}\left(a_i - \mu_i\right) + c_j - \sum_i w_{ij}\nabla\mu_i\right),\end{aligned}$$

showing that the unit's output does not change when changing the offset μ_i to $\tilde{\mu}_i$ if the unit's bias parameter c_j is reparameterized to $\tilde{c}_j = c_j + \sum_i w_{ij}\nabla\mu_i$. This generalizes the reparameterization for RBMs given by Equations (10) – (12) to ANNs. Now, by setting μ_i or $\tilde{\mu}_i$ to zero it follows that for each normal ANN there exists a centered ANN and *vice versa* such that the output of each neuron and thus the functional mapping from input to output of the whole network stays the same. This holds independently of the chosen activation functions, loss function and connection types including directed, undirected and recurrent connections.

Moreover, if we guarantee that a shift of a_i implies a shift of μ_i by the same value (that is, a shift of a_i to $a_i + \delta_i$ implies a shift of μ_i to $\mu_i + \delta_i$) the neuron's output o_j gets invariant to shifts of a_i. This is easy to see since δ_i cancels out in Equation (16) if the same shift is applied to both a_i and μ_i, which holds for example if we set the offsets to the mean values of the corresponding variables since $\langle a_i + \delta_i \rangle = \langle a_i \rangle + \delta_i$.

Note that, the original centering algorithm (LeCun et al., 1998; Schraudolph, 1998) did not reparameterize the network, which can cause instabilities especially if the learning rate is large.

4.1 Auto Encoders

An AE or auto-associator (Rumelhart et al., 1986b) is a type of ANN (and can thus be centered) that has originally been proposed for unsupervised dimensionality reduction. Like RBMs, AEs have also been used for unsupervised feature extraction and greedy layer-wise pre-training of deep neural networks (Bengio et al., 2007), and therefore fit well in this study for analyzing centering in ANNs.

In general, an AE consists of a deterministic encoder $encode(\mathbf{x})$, which maps the input $\mathbf{x} = (x_1, ..., x_N)$ to a hidden representation $\mathbf{h} = (h_1, ..., h_M)$ and a deterministic decoder $decode(\mathbf{h})$, which maps the hidden representation to the reconstructed input representation $\tilde{\mathbf{x}}$. The network is optimized such that the reconstructed input $\tilde{\mathbf{x}}$ gets as close as possible to the original input $\mathbf{x}$ measured by a chosen loss $\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}})$. Common choices for the loss are the mean squared error $\langle \sum_{i=1}^N (x_i - \tilde{x}_i)^2 \rangle_d$ for arbitrary input and the average cross entropy $\langle - \sum_{i=1}^N x_i \log \tilde{x}_i + (1 - x_i) \log(1 - \tilde{x}_i) \rangle_d$ for binary data. AEs are usually trained via back-propagation (Kelley, 1960; Rumelhart et al., 1986a) and they can be seen as feed-forward neural networks where the input patterns are also the desired output patterns. We can therefore define a centered AE by centering the encoder and decoder, which for a centered three layer AE corresponds to

$$\begin{aligned} encode(\mathbf{x}) &= \phi^{enc}\left(\mathbf{W}'(\mathbf{x} - \boldsymbol{\mu}) + \mathbf{c}\right) &= \mathbf{h}, \\ decode(\mathbf{h}) &= \phi^{dec}\left(\mathbf{W}(\mathbf{h} - \boldsymbol{\lambda}) + \mathbf{b}\right) &= \tilde{\mathbf{x}}, \end{aligned}$$

with encoder matrix $\mathbf{W}'$, decoder matrix $\mathbf{W}$, encoder bias $\mathbf{c}$, decoder bias $\mathbf{b}$, encoder offset $\boldsymbol{\mu}$, decoder offset $\boldsymbol{\lambda}$, encoder activation function ϕ^{enc} and decoder activation function ϕ^{dec}. It is common to assume tied weights, which means that the encoder matrix is just the transpose of the decoder matrix ($\mathbf{W}' = \mathbf{W}^T$). Although this reduces the number of free parameters AEs can still learn trivial representations if the number of hidden units is not chosen to be much smaller than the number of input dimensions or if the network is not regularized. Common regularized variants of AEs are denoising AEs (Vincent et al., 2008), sparse AEs (Olshausen et al., 1996; Poultney et al., 2006; Ng, 2011) and contractive AEs (Bishop, 1995; Rifai et al., 2011).

When choosing the activation functions for the encoder and decoder (that are sigmoid, tangens-hyperbolicus, radial-basis, linear, linear-rectifier, ...), we have to ensure that the encoder activation function is appropriate for the input data (for example a sigmoid cannot represent negative values). It is worth mentioning that when using the sigmoid function for ϕ^{enc} and ϕ^{dec} and tied weights, the encoder becomes equivalent to Equation (5) and the decoder becomes equivalent to Equation (6). The network structure therefore becomes equivalent to an RBM such that the only difference is the training objective (that is the loss function).

4.2 Initialization of the Model Parameters

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

Hinton (2010) has proposed to initialize the RBM's 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 $p_i = \langle x_i \rangle_d$). He has stated 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

$$\mathbf{b}^* = \ln\left(\frac{\langle \mathbf{x} \rangle_d}{\mathbf{1} - \langle \mathbf{x} \rangle_d}\right) = -\ln\left(\frac{\mathbf{1}}{\langle \mathbf{x} \rangle_d} - \mathbf{1}\right) = \sigma^{-1}(\langle \mathbf{x} \rangle_d) \ , \tag{17}$$

where σ^{-1} is the inverse sigmoid function. Note 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 of an RBM with this initialization is then given by $p(\mathbf{x} = \mathbf{1}|\mathbf{h}) = \sigma(\sigma^{-1}(\langle \mathbf{x} \rangle_d)) = \langle \mathbf{x} \rangle_d$, where $p(\mathbf{x} = \mathbf{1}|\mathbf{h})$ denotes the vector containing the elements $p(x_i = 1|\mathbf{h})$. 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. For bigger initial weight values the model can be initially centered by using the inverse sigmoid of the average hidden activity given the data ($\mathbf{c}^* = \sigma^{-1}(\langle p(\mathbf{h} = \mathbf{1}|\mathbf{x}) \rangle_d)$) for the hidden biases.

Montavon and Müller (2012) have suggested to initialize the bias parameters of centered DBMs to the inverse sigmoid of the initial offset parameters. They have argued 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 $\mu_i = \langle x_i \rangle_d$ 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 (17).

Note that this initialization is not restricted to RBMs or the sigmoid activation function. Independently of the initial weight matrix we can always set the bias in ANNs to the inverse activation function of the corresponding mean value.

5. Methods

As shown in Section 3 the algorithms described by Cho et al. (2011), Tang and Sutskever (2011) and Montavon and Müller (2012) 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 Section 5 we therefore empirically analyze the different ways of centering and try to achieve a deeper understanding of why centering is beneficial.

For simplicity we introduce the following shorthand notation. The letter d denotes the data mean $\langle\cdot\rangle_d$, m denotes the model mean $\langle\cdot\rangle_m$, a denotes the average of the means $\frac{1}{2}\langle\cdot\rangle_d + \frac{1}{2}\langle\cdot\rangle_m$, and *0* is used if the offsets are 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 $\boldsymbol{\mu} = \langle\mathbf{x}\rangle_d$ and $\boldsymbol{\lambda} = \langle\mathbf{h}\rangle_m$. We add a superscribed b (before) or l (later/after) to denote whether the reparameterization is performed before or after the gradient update. If the sliding factor in Algorithm 1 or 2 is set to a value smaller than one and thus an exponentially moving average is used, a subscript s (sliding) is added. Thus, we represent the variant of Cho

Abbr.	μ	λ	Description
00	$\mathbf{0}$	$\mathbf{0}$	Normal binary RBM (Smolensky, 1986)
d0	$\langle\mathbf{x}\rangle_d$	$\mathbf{0}$	Data Normalization RBM (Tang and Sutskever, 2011)
dd_s^l	$\langle\mathbf{x}\rangle_d$	$\langle\mathbf{h}\rangle_d$	Original Centered RBM (Montavon and Müller, 2012) reparam. after gradient update, use of an exp. moving average
aa^b	$0.5\,(\langle\mathbf{x}\rangle_d + \langle\mathbf{x}\rangle_m)$	$0.5\,(\langle\mathbf{h}\rangle_d + \langle\mathbf{h}\rangle_m)$	Enhanced gradient RBM (Cho et al., 2011) reparam. before gradient update, no use of an exp. moving average
dd_s^b	$\langle\mathbf{x}\rangle_d$	$\langle\mathbf{h}\rangle_d$	Centering using the data mean, reparam. before gradient update, use of an exp. moving average
mm_s^b	$\langle\mathbf{x}\rangle_m$	$\langle\mathbf{h}\rangle_m$	Centering using the model mean, reparam. before gradient update, use of an exp. moving average
dm_s^b	$\langle\mathbf{x}\rangle_d$	$\langle\mathbf{h}\rangle_m$	Centering using the data mean for the visible and the model mean for hidden units, reparam. before gradient update, use of an exp. moving average

Table 1: Look-up table: Abbreviations for the most frequently used algorithms.

et al. (2011) by aa^b, the one of Montavon and Müller (2012) by dd^l_s, the data normalization of Tang and Sutskever (2011) by *d0*, and the normal binary RBM simply by *00*. Table 1 summarizes the abbreviations most frequently used in this paper.

We begin our analysis with experiments on 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 initializations described in Section 4.2. 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. We then compare the normal and the centered gradient with the natural gradient. Finally, to verify whether the results scale to more realistic problem sizes we compare RBMs, DBMs and AE on ten large data sets.

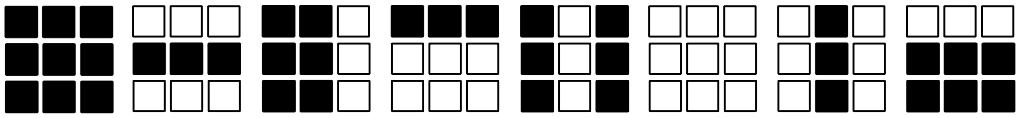
5.1 Benchmark Problems

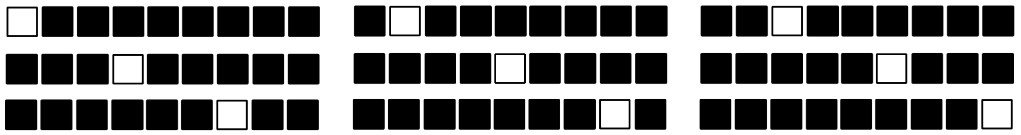
We consider twelve different benchmark problems in our detailed analysis, see also Figure 2. The ***Bars & Stripes*** (MacKay, 2003) data set consists of patterns of size $D \times D$ that can be generated as follows. First, for each row of a pattern all corresponding pixels are either set to zero or to one with equal probability. Second, with a probability of 0.5 the pattern is rotated by 90 degrees. 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 6.9) leading to an upper bound of -41.59 and -13.86, respectively.

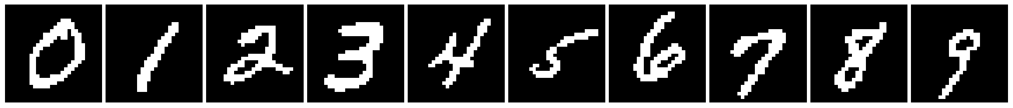
The ***Shifting Bar*** data set is an artificial benchmark problem we have designed to be asymmetric in terms of the amount of zeros and ones in the data. The data set consists of vectors of size N where a set of B successive pixels with cyclic boundary conditions are set to one, the 'bar', and the others are set to zero. More formally the data set can be generated as follows, beginning with a vector where all pixels are set to zero, an index $0 \leq i < N$ is chosen uniformly at random. The pixel with index i and the following $B-1$ pixels, which underly cyclic boundary conditions are set to one (that is the pixels with indices $\{i,\ (i+1) \mod N\ , \dots\ , (i+B-1) \mod N\}$ are set to one). This leads to N different patterns with equal probability, and an upper bound of the LL of $N\ln\left(\frac{1}{N}\right)$, the percentage of ones in the data set is $\frac{B}{N}$. 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., 1998) data set 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. Usually, in a first preprocessing step the values are normalized to lie in $[0,1]$. Across different studies the normalized values are then either used directly as input, or the data set is binarized using a threshold of 0.5 or by sampling according to the normalized values that are treated as probabilities.[4] In this study

4. In Appendix D we give a an comparison of normal and centered RBMs on the different *MNIST* versions.


(a) 8 out of 16 patterns from the *Bars & Stripes* data set.


(b) All patterns of the *Shifting Bar* data set with $N = 9$ and $B = 1$.


(c) Example patterns from the *MNIST* data set after binarization.

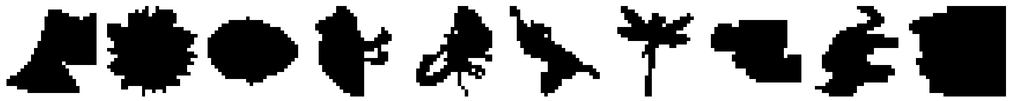

(d) Example patterns from the *Caltech 101 Silhouette* data set.

Figure 2: Some patterns from the different benchmark problems.

we binarized *MNIST* 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***.

The ***CalTech 101 Silhouettes*** (Marlin et al., 2010) data set consists of 4100 training, 2307 validation, and 2264 testing examples of binary object silhouettes of size 28×28. The data set contains 55.1% ones, and thus (like in the *Bars & Stripes problem*) the amount of zeros and ones is almost the same. The background pixels take the value one, which is in contrast to *MNIST* where the background pixels are set to zero.

In some experiments we also considered the eight ***UCI binary*** (Larochelle et al., 2010; Larochelle and Murray, 2011) data sets from different domains comprising biological, image, text and game-related data. The data sets differ in dimensionality (112 to 500) and size (a few hundred to several thousand examples) and have been separated into training, validation, and test sets. All *UCI binary* data sets contain less ones than zeros, where the percentage of ones lies between 3.9% and 36.8%.

6. Results

For all models in this work the 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 biases, hidden biases, and offsets were initialized as described in Section 4.2.

We began our analysis with experiments on small RBMs where the LL can be calculated exactly, where we used 4 hidden units when modeling *Bars & Stripes* and *Shifting Bar* and 16 hidden units when modeling *MNIST*. For training we used CD-1, PCD-1 and PT$_c$ (with $c = 10$ or $c = 20$) where the c temperatures were distributed uniformly from 0 to 1. For *Bars & Stripes* and *Shifting Bar* full-batch training was performed for $50,000$ gradient updates, where the LL was evaluated every 50th gradient update. For modeling *MNIST* mini-batch training with a batch size of 100 was performed for 100 epochs, each consisting of 600 gradient updates and the exact LL was evaluated after each epoch. Note that in order to get an unbiased comparison of the different models, we did not use any additional modifications of the update rule like a momentum term, weight decay or an annealing learning rate.

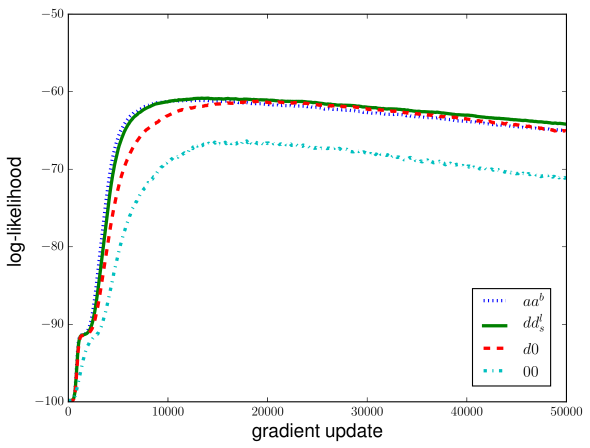
The tables in this work show the maximum average LL and the corresponding standard deviation reached during training averaged over 25 trials. In addition the final average LL reached at the end of training is given in parenthesis to indicate a potential divergence of the LL. For some computational expensive experiments we used only 10 trials, which will be mentioned explicitly. For reasons of readability, the LL values for the big data sets were normalized to the number of training samples. 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.

6.1 Comparison of the Standard Methods

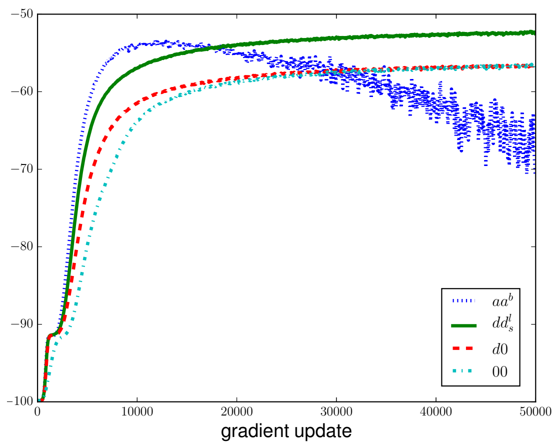
The comparison of the learning performance of the previously described algorithms dd^l_s, aa^b, *d0*, and *00* (using their originally proposed initializations) shows that training a centered RBM leads to significantly higher LL values than training a normal binary RBM (see Table 2 for the results on *Bars & Stripes* and *MNIST* and Table 3 for the results on *Shifting Bar* and *Flipped Shifting Bar*).

Figure 3 illustrates on the *Bars & Stripes* data set that centering (aa^b, dd^l_s, and $d0$) leads to faster learning and higher LL values than normal RBMs (*00*). This differs from the observations made for DBMs by Montavon and Müller (2012), who have found a better generative performance of centering only in the case of locally connected DBMs. Furthermore, Figure 3 shows that centering both the visible and the hidden variables (dd^l_s and aa^b) compared to centering only the visible variables (*d0*) accelerates learning and leads to a higher LL. This is different from the observations made for RBMs by Tang and Sutskever (2011), who have stated that centering only the visible variables leads to a performance similar to that of the enhanced gradient. Thus centered RBMs where visible and hidden variables are centered can form more accurate models of the data distribution than normal RBMs and centered RBMs where only the visible variables are centered.

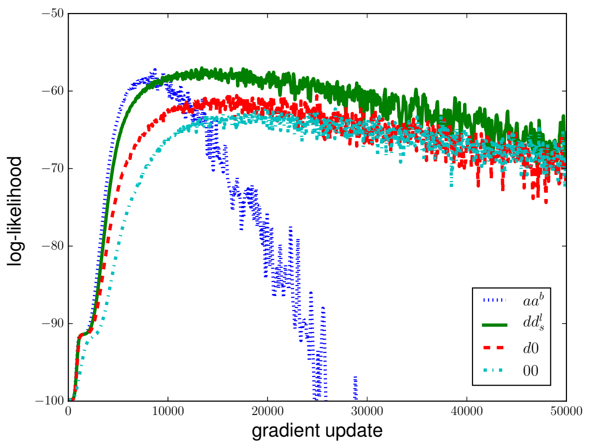
It can also be seen from Figure 3 that all methods show divergence in combination with CD and PCD (as described before by Fischer and Igel, 2010, for normal RBMs), which is prevented for dd^l_s, *d0*, and *00* when using PT as shown in Figure 3(b). This can be explained by the fact that PT leads to faster mixing Markov chains and thus less biased gradient approximations. The *aa* algorithm however suffers from severe divergence of the LL when PCD or PT is used, which is even worse than with CD. This divergence problem does occur independently of the choice of the learning rate as indicated by the LL values reached at the end of training (given in parentheses) in Table 2 and Table 3, and which can



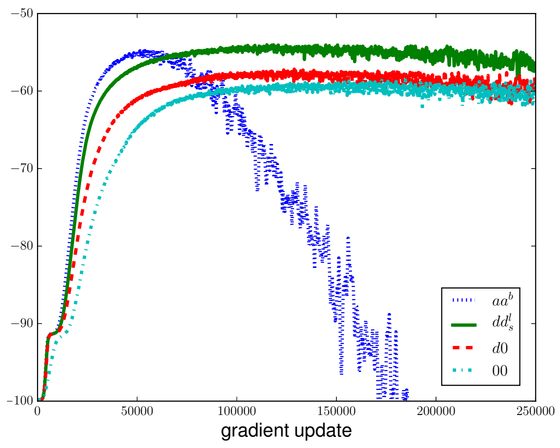
(a) CD-1 - learning rate 0.05



(b) PT_{10} - learning rate 0.05



(c) PCD-1 - learning rate 0.05



(d) PCD-1 - learning rate 0.01

Figure 3: Evolution of the average LL on the training data for models with 4 hidden units on the *Bars & Stripes* data set for the standard centering methods. (a) CD-1 with a learning rate of $\eta = 0.05$, (b) PT_{10} with a learning rate of $\eta = 0.05$, (c) PCD-1 with a learning rate of $\eta = 0.05$, and (d) PCD-1 with a learning rate of $\eta = 0.01$.

Algorithm-η	aa^b	dd^l_s	$d0$	00
Bars & Stripes				
CD-1-0.1	**-61.66** ±1.71 (-69.1)	**-61.33** ±1.89 (-69.1)	**-61.73** ±4.19 (-70.9)	-66.53 ±3.47 (-78.1)
CD-1-0.05	**-61.09** ±1.89 (-65.0)	**-60.82** ±2.31 (-64.2)	**-61.23** ±3.70 (-65.1)	-66.35 ±4.24 (-71.2)
CD-1-0.01	**-61.08** ±1.60 (-61.1)	**-61.31** ±1.55 (-61.3)	-63.30 ±3.02 (-63.3)	-68.56 ±2.94 (-68.6)
PCD-1-0.1	**-59.05** ±2.07 (-360.6)	**-58.93** ±2.24 (-102.7)	-62.46 ±4.65 (-97.3)	-63.28 ±5.61 (-84.3)
PCD-1-0.05	**-57.25** ±1.51 (-167.4)	**-57.01** ±1.61 (-65.4)	-60.42 ±5.19 (-72.5)	-62.29 ±4.84 (-70.6)
PCD-1-0.01	**-54.86** ±0.98 (-55.3)	-56.80 ±0.73 (-56.8)	-61.03 ±3.74 (-61.0)	-64.61 ±2.95 (-64.6)
PT_{10}-0.1	-54.67 ±3.43 (-202.5)	**-51.99** ±1.13 (-52.2)	-56.18 ±5.32 (-56.7)	-54.89 ±3.66 (-55.3)
PT_{10}-0.05	-53.34 ±0.99 (-70.7)	**-52.21** ±1.06 (-52.5)	-56.54 ±5.82 (-56.6)	-56.46 ±4.43 (-56.8)
PT_{10}-0.01	**-53.63** ±1.25 (-53.8)	-56.76 ±0.79 (-56.8)	-61.26 ±4.58 (-61.3)	-64.71 ±3.52 (-64.7)
MNIST				
CD-1-0.1	**-153.01** ±2.32 (-158.0)	**-153.46** ±2.45 (-155.5)	**-153.89** ±2.91 (-155.0)	-170.29 ±2.45 (-170.6)
CD-1-0.05	**-153.07** ±1.51 (-156.0)	**-152.78** ±3.57 (-153.8)	**-153.01** ±2.80 (-155.4)	-169.43 ±2.95 (-169.7)
CD-1-0.01	**-152.90** ±1.38 (-152.9)	**-152.42** ±2.15 (-152.4)	-153.77 ±2.54 (-153.8)	-172.70 ±1.86 (-172.8)
PCD-1-0.1	-150.78 ±1.95 (-174.9)	**-144.10** ±1.67 (-144.7)	-154.64 ±3.14 (-156.6)	-183.11 ±8.62 (-183.2)
PCD-1-0.05	-146.83 ±1.54 (-163.4)	**-141.93** ±1.25 (-142.9)	-146.86 ±2.08 (-147.5)	-185.78 ±6.78 (-185.8)
PCD-1-0.01	-143.36 ±0.74 (-144.1)	**-140.43** ±0.62 (-140.7)	-141.88 ±1.02 (-141.9)	-193.60 ±4.56 (-193.6)
PT_{10}-0.01	-259.50 ±17.54 (<-999)	**-142.00** ±1.04 (-142.3)	-145.84 ±2.49 (-147.4)	-152.38 ±3.05 (-152.4)

Table 2: Maximum average LL on the training data for centered and normal RBMs (top) with 4 hidden units on the *Bars & Stripes* data set and (bottom) with 16 hidden units on the *MNIST* data set using different sampling methods and learning rates η. Each LL value is followed by the corresponding standard deviation of the 25 trials and the average LL at the end of training is given in parenthesis.

Algorithm-η	aa^b	dd^l_s	$d0$	00
Shifting Bar				
CD-1-0.2	-21.04 ±1.41 (-21.9)	**-20.42** ±0.85 (-20.4)	-22.15 ±1.38 (-22.5)	-22.32 ±1.46 (-22.6)
CD-1-0.1	-21.29 ±1.18 (-21.5)	**-20.76** ±0.80 (-20.8)	-21.35 ±0.95 (-21.4)	-21.56 ±0.94 (-21.6)
CD-1-0.05	**-21.16** ±0.84 (-21.2)	-22.74 ±0.65 (-22.7)	-26.89 ±0.29 (-26.9)	-26.11 ±0.40 (-26.1)
PCD-1-0.2	-23.07 ±0.78 (-237.2)	**-22.48** ±0.85 (-33.6)	-23.15 ±1.01 (-31.9)	-23.34 ±0.75 (-31.7)
PCD-1-0.1	**-22.14** ±0.70 (-87.4)	**-21.92** ±0.66 (-24.0)	-22.75 ±0.84 (-23.7)	-22.64 ±1.15 (-23.3)
PCD-1-0.05	**-21.71** ±0.73 (-26.0)	-22.53 ±0.55 (-22.5)	-26.84 ±0.36 (-26.8)	-26.06 ±0.48 (-26.1)
PT$_{10}$-0.2	-21.30 ±0.81 (-31.9)	**-20.66** ±0.67 (-20.8)	-21.47 ±1.13 (-21.6)	-22.12 ±1.17 (-22.3)
PT$_{10}$-0.1	-20.84 ±0.62 (-21.5)	**-20.57** ±0.56 (-20.6)	-21.34 ±0.90 (-21.4)	-21.21 ±0.93 (-21.2)
PT$_{10}$-0.05	**-20.78** ±0.88 (-20.8)	-22.42 ±0.69 (-22.4)	-26.96 ±0.30 (-27.0)	-26.18 ±0.38 (-26.2)
Flipped Shifting Bar				
CD-1-0.2	**-20.82** ±1.15 (-21.3)	**-20.73** ±1.06 (-20.8)	-22.04 ±1.62 (-22.3)	-28.14 ±0.27 (-28.2)
CD-1-0.1	**-20.85** ±1.05 (-20.9)	**-20.90** ±0.83 (-20.9)	-21.17 ±0.84 (-21.2)	-28.35 ±0.04 (-28.4)
CD-1-0.05	**-21.18** ±0.82 (-21.2)	-22.64 ±0.65 (-22.6)	-26.85 ±0.34 (-26.9)	-28.31 ±0.02 (-28.3)
PCD-1-0.2	-22.73 ±0.88 (-310.8)	**-22.35** ±0.91 (-29.1)	-23.48 ±0.84 (-32.6)	-28.19 ±0.28 (-28.3)
PCD-1-0.1	-22.15 ±0.71 (-88.3)	**-21.64** ±0.72 (-22.6)	-22.38 ±0.88 (-23.2)	-28.35 ±0.04 (-28.4)
PCD-1-0.05	**-21.72** ±0.86 (-25.6)	-22.34 ±0.60 (-22.3)	-26.90 ±0.34 (-26.9)	-28.31 ±0.02 (-28.3)
PT$_{10}$-0.2	-21.13 ±0.72 (-32.4)	**-20.57** ±0.63 (-20.6)	-21.14 ±0.84 (-21.4)	-28.17 ±0.26 (-28.2)
PT$_{10}$-0.1	-21.03 ±0.81 (-21.6)	**-20.73** ±0.74 (-20.8)	-21.26 ±0.72 (-21.3)	-28.35 ±0.04 (-28.4)
PT$_{10}$-0.05	**-21.01** ±0.86 (-21.1)	-22.48 ±0.73 (-22.5)	-26.87 ±0.35 (-26.9)	-28.31 ±0.02 (-28.3)

Table 3: Maximum average LL on the training data for centered and normal RBMs with 4 hidden units on (top) the *Shifting Bar* data set and (bottom) the *Flipped Shifting Bar* data set using different sampling methods and learning rates η. Each LL value is followed by the corresponding standard deviation of the 25 trials and the average LL at the end of training is given in parenthesis.

also be seen by comparing Figure 3(c) and Figure 3(d). The divergence occurs earlier and is more severe for bigger learning rates, while for the other algorithms we never observed divergence in combination with PT even for very big learning rates and long training time. The reasons for this divergence are discussed in detail in Section 6.4.

The results in Table 3 also demonstrate the flip invariance of the centered RBMs on the *Shifting Bar* data set empirically. Whereas *00* fails to model the flipped version of the data set correctly, dd^l_s, aa^b, and *d0* have approximately the same performance on the flipped and unflipped data set.

6.2 Initialization

The experiments in this section were done to analyze the effects of different initializations of the parameters as discussed in Section 4.2. 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 4 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 better performance and therefore seem to be less sensitive

Algorithm-η	00 *init zero*			00 *init* σ^{-1}		
Flipped Shifting Bar						
CD-1-0.2	-28.14	±0.27	(-28.2)	**-22.08**	±1.49	(-22.5)
CD-1-0.1	-28.35	±0.04	(-28.4)	**-21.40**	±1.21	(-21.6)
CD-1-0.05	-28.31	±0.02	(-28.3)	**-24.90**	±0.47	(-24.9)
PCD-1-0.2	-28.19	±0.28	(-28.3)	**-25.17**	±1.32	(-42.3)
PCD-1-0.1	-28.35	±0.04	(-28.4)	**-23.71**	±0.98	(-26.7)
PCD-1-0.05	-28.31	±0.02	(-28.3)	**-24.99**	±0.57	(-25.0)
PT$_{10}$-0.2	-28.17	±0.26	(-28.2)	**-22.83**	±1.41	(-23.5)
PT$_{10}$-0.1	-28.35	±0.04	(-28.4)	**-21.52**	±0.88	(-21.8)
PT$_{10}$-0.05	-28.31	±0.02	(-28.3)	**-24.87**	±0.50	(-24.9)
MNIST						
CD-1-0.1	**-170.29**	±2.45	(-170.6)	**-169.62**	±4.45	(-169.8)
CD-1-0.05	**-169.43**	±2.95	(-169.7)	-171.12	±3.05	(-171.2)
CD-1-0.01	-172.70	±1.86	(-172.8)	**-169.97**	±2.02	(-172.3)
PCD-1-0.1	-183.11	±8.62	(-183.2)	**-164.68**	±3.27	(-189.7)
PCD-1-0.05	-185.78	±6.78	(-185.8)	**-156.93**	±4.10	(-158.3)
PCD-1-0.01	-193.60	±4.56	(-193.6)	**-144.28**	±0.93	(-144.3)
PT$_{10}$-0.01	-152.38	±3.05	(-152.4)	**-148.79**	±2.44	(-150.7)

Table 4: Maximum average LL on the training data for normal RBMs (top) with 4 hidden units on the *Flipped Shifting Bar* data set and (bottom) with 16 hidden units 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^l_s *init zero*			dd^l_s *init* σ^{-1}		
FLIPPED SHIFTING BAR						
CD-1-0.2	**-20.76**	±0.95	(-20.8)	**-20.73**	±1.06	(-20.8)
CD-1-0.1	**-20.82**	±0.88	(-20.8)	**-20.90**	±0.83	(-20.9)
CD-1-0.05	**-22.98**	±0.73	(-23.0)	**-22.64**	±0.65	(-22.6)
PCD-1-0.2	**-22.32**	±0.82	(-31.8)	**-22.35**	±0.91	(-29.1)
PCD-1-0.1	**-21.75**	±0.60	(-22.6)	**-21.64**	±0.72	(-22.6)
PCD-1-0.05	-22.79	±0.64	(-22.8)	**-22.34**	±0.60	(-22.3)
PT_{10}-0.2	**-20.37**	±0.36	(-20.6)	**-20.57**	±0.63	(-20.6)
PT_{10}-0.1	**-20.74**	±0.75	(-20.8)	**-20.73**	±0.74	(-20.8)
PT_{10}-0.05	-22.95	±0.70	(-22.9)	**-22.48**	±0.73	(-22.5)

Table 5: Maximum average LL on the training data for centered RBMs (dd^l_s) with 4 hidden units on the *Flipped Shifting Bar* data set, where the visible bias is initialized to zero or to the inverse sigmoid of the data mean.

to the 'difficulty' of the data representation. However, the results are not as good as the results of the centered RBMs shown in Table 3. The same observations can be made when training RBMs on the *MNIST* data set (see Table 4 bottom). The RBMs with inverse sigmoid initialization achieved significantly better results than RBMs initialized to zero in the case of PCD and PT, but still worse compared to the centered RBMs. Furthermore, using the inverse sigmoid initialization allows us to achieve similar performance on *MNIST* and *1-MNIST*, while the RBM with zero initialization failed to learn the distribution for *1-MNIST* at all (results not shown). A visual comparison of the filters of normal RBMs trained on *MNIST* and *1-MNIST* has already been given by Cho et al. (2011) and Tang and Sutskever (2011).

We then trained models using the centering versions *dd*, *aa*, and *d0* comparing the initialization suggested in Section 4.2 against the initialization to zero. In most cases the results showed no significant difference in terms of the maximum LL reached in trials with different initializations, or slightly better results were found when using the inverse sigmoid in combination with a small learning rate. This can be explained by the better starting point yielded by this initialization. See Table 5 for the results for dd^l_s on the *Flipped Shifting Bar* data set as an example. We therefore used the inverse sigmoid initialization in the following experiments since it was beneficial for normal RBMs.

6.3 Reparameterization

To investigate the effects of performing the reparameterization before or after the gradient update during training of centered RBMs , we analyzed the learning behavior of dd^b_s (the algorithm suggested here) and dd^l_s (the algorithm suggested by Montavon and Müller, 2012) on all data sets. The results for RBMs trained on the *Bars & Stripes* data set and the *MNIST* data set are given in Table 6 (top). No significant difference between the two versions can be observed and the same result (not shown) was obtained for the *Shifting Bar*

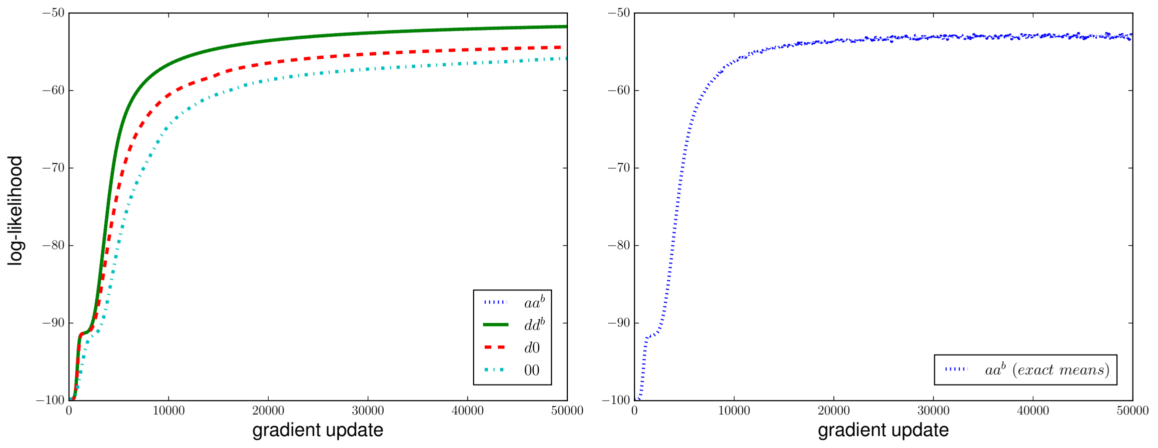
Algorithm-η	dd_s^b			dd_s^l		
Bars & Stripes						
CD-1-0.1	**-61.32**	±1.87	(-69.1)	**-61.33**	±1.89	(-69.1)
CD-1-0.05	**-60.82**	±2.26	(-64.2)	**-60.82**	±2.31	(-64.2)
CD-1-0.01	**-61.31**	±1.55	(-61.3)	**-61.31**	±1.55	(-61.3)
PCD-1-0.1	**-59.08**	±1.53	(-94.5)	**-58.93**	±2.24	(-102.7)
PCD-1-0.05	**-56.97**	±1.74	(-64.3)	**-57.01**	±1.61	(-65.4)
PCD-1-0.01	**-56.88**	±0.75	(-56.9)	**-56.80**	±0.73	(-56.8)
PT_{10}-0.1	**-51.99**	±1.29	(-52.5)	**-51.99**	±1.13	(-52.2)
PT_{10}-0.05	**-52.34**	±1.15	(-52.5)	**-52.21**	±1.06	(-52.5)
PT_{10}-0.01	**-56.76**	±0.77	(-56.8)	**-56.76**	±0.79	(-56.8)
MNIST						
CD-1-0.1	**-153.16**	±2.57	(-154.9)	**-153.46**	±2.45	(-155.5)
CD-1-0.05	**-152.34**	±3.61	(-153.1)	**-152.78**	±3.57	(-153.8)
CD-1-0.01	**-152.23**	±2.03	(-152.2)	**-152.42**	±2.15	(-152.4)
PCD-1-0.1	**-144.12**	±2.12	(-145.2)	**-144.10**	±1.67	(-144.7)
PCD-1-0.05	**-141.88**	±1.28	(-142.2)	**-141.93**	±1.25	(-142.9)
PCD-1-0.01	**-140.40**	±0.66	(-140.7)	**-140.43**	±0.62	(-140.7)
PT_{10}-0.01	**-141.95**	±1.22	(-142.5)	**-142.00**	±1.04	(-142.3)

Table 6: Maximum average LL on the training data for RBMs (top) with 4 hidden units on the *Bars & Stripes* data set and (bottom) with 16 hidden units on the *MNIST* data set, using the reparameterization before (dd_s^b) and after (dd_s^l) the gradient update.

and the *Flipped Shifting Bar* data set. We therefore reparameterize the RBMs before the gradient update in the remainder of this work. This is also motivated by the fact that the enhanced gradient is only equivalent to centering using the average of model and data mean for visible and hidden offsets if the model is reparameterized before it is updated (that is, the enhanced gradient is equivalent to aa^b but not to aa^l).

6.4 Analyzing the Model-Mean-Related Divergence Effect

The severe divergence problem observed when using the enhanced gradient in combination with PCD or PT raises the question whether the problem is induced by setting the offsets to $0.5(\langle \mathbf{x} \rangle_d + \langle \mathbf{x} \rangle_m)$ and $0.5(\langle \mathbf{h} \rangle_d + \langle \mathbf{h} \rangle_m)$ or by bad sampling based estimates of gradient and offsets. We therefore trained centered RBMs with 9 visible and 4 hidden units on the *Bars & Stripes* data set using either the exact gradient where only $\langle \mathbf{x} \rangle_m$ and $\langle \mathbf{h} \rangle_m$ were approximated by samples or using PT_{10} estimates of the gradient while $\langle \mathbf{x} \rangle_m$ and $\langle \mathbf{h} \rangle_m$ were calculated exactly. Figure 4(a) shows that if the exact gradient is used but the offsets are estimated, no divergence for *aa* in combination with PT is observed and the performance of *aa* and *dd* become almost equivalent. Interestingly, the divergence is also prevented if the gradient is estimated using PT but the offsets are calculated exactly as shown in Figure 4(b).



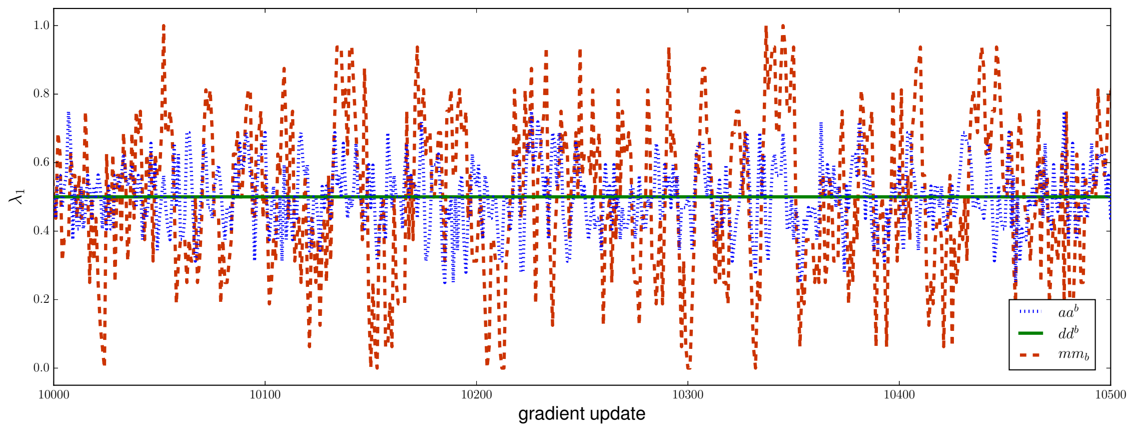
(a) Exact gradient with approximated offsets (b) PT_{10} with exact means

Figure 4: Evolution of the average LL on the training data for RBMs with 4 hidden units on the *Bars & Stripes* data set for the standard centering methods. (a) When the exact gradient is used but the offsets are estimated using PT_{10} and (b) when PT_{10} is used for estimating the gradient but the offsets are calculated exactly. In both cases the learning rate was $\eta = 0.05$.

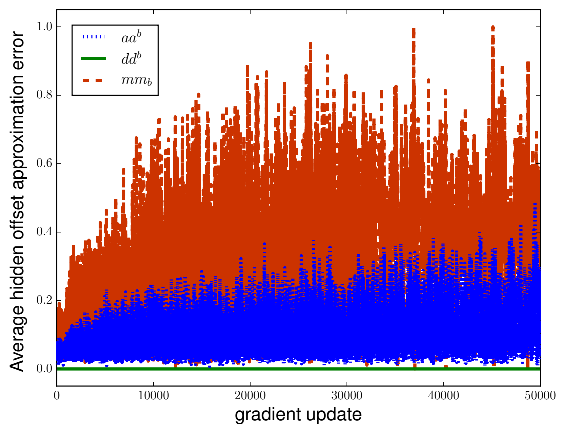
Thus, the divergence behavior must be induced by approximating both, gradient and offsets. The fact that the mean under the data distribution can always be calculated exactly might explain why we do not observe divergence for dd in combination with PT.

To further deepen the understanding of the divergence effect we investigated the parameter evolution during training of RBMs with different offsets. We observed that the change of the offset values between two gradient updates gets extremely large during training when using the model mean. Figure 5(a) shows in an exemplary manner the evolution of the first hidden offset λ_1 for a single trial, where the approximated offset for dd^b is almost constant while it is rather large for aa^b and even bigger for mm^b (the centering variant that uses the mean of hidden and visible variables under the model distribution as offsets). In each iteration we calculated the exact offsets to estimate the approximation error shown in Figure 5(b). Obviously, there is no approximation error for dd while the error for aa quickly gets large and the error for mm gets even twice as big. In combination with the gradient approximation error this causes the weight matrices for aa and mm to grow extremely big as shown in Figure 5(c). Consistent with the even larger approximation error for the offset estimate of mm, the divergence for mm becomes even worse than that for aa, as shown in Figure 6.

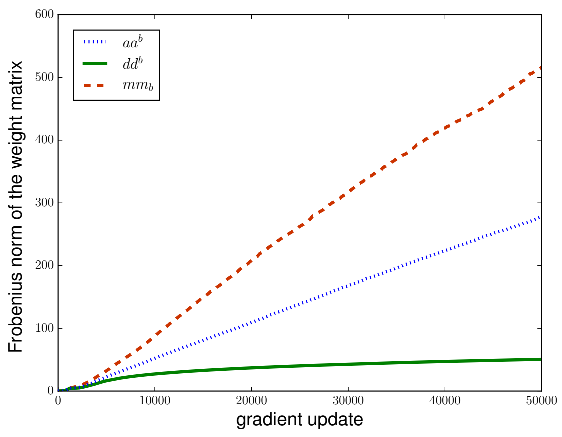
To further confirm that the divergence is not just caused by the additional sampling noise introduced by approximating the offsets, we trained a centered RBM using PT for the gradient approximation while we set the offsets to uniform random values between zero and one in each iteration. The results are shown in Figure 6(a) and demonstrate that even random offset values do not lead to the divergence problems. Thus, the divergence cannot be caused by additional sampling noise but rather by the correlated errors of gradient and offset approximations. To test this hypothesis we investigated mm^b where the samples for offset and gradient approximations were taken from different PT sampling processes. The results are shown in Figure 6(b) where no divergence can be observed anymore. While creating two sets of samples for gradient and offset approximations prevents the LL from diverging it almost doubles the computational cost and can therefore not be considered as a reasonable solution in practice. Moreover, using the model mean as offset still leads to



(a) Close up of the offset evolution for λ_1

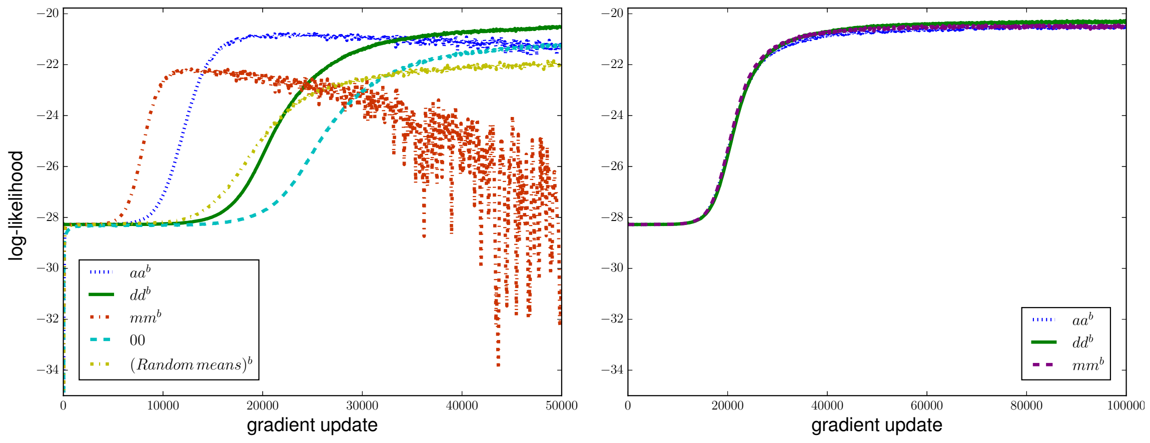


(b) Evolution of the offset approximation error



(c) Evolution of the weight norm

Figure 5: Evolution of the offsets and weights of different centered variants for RBMs with 4 hidden units during training on *Bars & Stripes* using PT_{10}. For clearness a single trial is shown, but the experiments were repeated 25 times and all trials showed qualitatively the same results. (a) Close up of the evolution of offset λ_1 over 500 gradient updates. (b) The evolution of the absolute difference between exact and approximated hidden offset averaged over all hidden units. (c) The evolution of the Frobenius norm of the weight matrices.



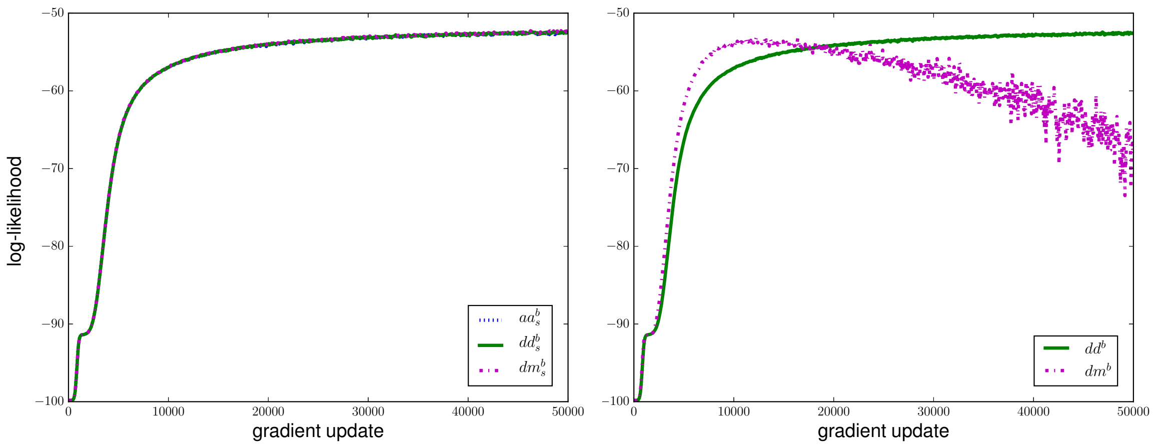
(a) Random offsets (b) Independent approximations

Figure 6: Evolution of the average LL on the training data for RBMs with 4 hidden units on the *Shifting Bars* data set using PT$_{10}$ with a learning rate of $\eta = 0.1$. (a) Normal RBMs, centered RBMs, and centered RBMs with random offset values (denoted by *Random means*^{b}). (b) Centered RBMs when the samples for the offset approximations come from a different Markov chain than the samples used for the gradient approximation.

slightly worse final LL values than using the mean under the data distribution. This might be explained by the fact that the additional approximation of the model mean introduces noise while the data mean can be estimated exactly. Note, that the divergence also occurs if either only visible or hidden offsets are set to the PT-approximated model mean, which can be seen for example for dm in Figure 7(b).

Interestingly, the observed initially faster learning speed of mm and aa, which can be seen in Figure 6(a), does not occur anymore when offset and gradient approximation are based on different sample sets. This observation can also be made when the exact gradient is used (see Figure 4a). Thus, the initially faster learning speed seems also be caused by the correlated approximations of gradient and offsets and could be explained by the rapidly growing norm of the weight matrix shown in Figure 5(c), which effectively leads to a bigger step size in the parameter updates.

Note, that divergence can either be caused by using the expectation under the model distribution for the offset approximations as described in this section or by the bias of the gradient approximation introduced by using only a few steps of Gibbs sampling (Fischer and Igel, 2010; Schulz et al., 2010; Fischer and Igel, 2011). The latter can occur for all methods when CD or PCD is used for training. Furthermore, the LL can also decrease on the test data due to overfitting to the training data, which is not divergence in the proper sense.



(a) With exponentially moving average (b) Without exponentially moving average

Figure 7: Evolution of the average LL on the training data for RBMs with 4 hidden units on *Bars & Stripes* with different centering variants, using PT_{10} and a learning rate of $\eta = 0.05$. (a) Using an exponentially moving average with a sliding factor of 0.01 (the curves are almost equivalent) and (b) without exponentially moving average (see also Figure 3b for aa^b)

6.5 Usage of an Exponentially Moving Average

The approximation of the mean values of the variables using just a few samples might be too inaccurate or biased by the samples in the current batch. This is in particular the case when the model mean is used as shown in Section 6.4, but also when the data mean is used with a small batch size. Therefore, an exponentially moving average can be used to smooth the approximation of the offsets between parameter updates, which stabilizes the approximations when small batch sizes are used as well as when the model mean is used for the offsets.

We analyzed the impact of using an exponentially moving average with a sliding factor of 0.01 for the estimation of the offset parameters. Figure 7(a) illustrates on the *Bars& Stripes* data set that the learning curves of the different models become almost equivalent and that the divergence problem when using the model mean does not occur anymore when an exponentially moving average is used. However, the maximum LL values reached are the same whether an exponentially moving average is used or not. This can be seen by comparing Figure 7(a) and Figure 7(b) and also by comparing the results in Table 2 and Table 3 with those in Table 7. As discussed in Section 6.4, this problem is caused by the correlation between the approximation error of gradient and offsets. When using an exponential moving average the current offsets contain only a small fraction of the current mean such that the correlation is highly reduced.

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 7(b) for example or in the case of mini-batch learning (results not shown). Thus, *dd* seems to be generally more stable than the other centering variants, which is also true for big models as will be shown at the end of Section 6.7.

In the previous experiments an exponentially moving average was used for the approximation of visible and hidden offsets as originally suggested by Montavon and Müller (2012). Note however, that in batch learning when $\langle \mathbf{x} \rangle_d$ is used for the visible offsets, these values stay constant such that an exponentially moving average has no effect. More generally

Algorithm-η	aa_s^b	dd_s^b	dm_s^b
Bars & Stripes			
CD-1-0.1	**-60.90** ±2.14 (-70.5)	**-61.32** ±1.87 (-69.1)	**-61.09** ±2.18 (-68.8)
CD-1-0.05	**-60.25** ±2.62 (-64.2)	**-60.82** ±2.26 (-64.2)	**-60.83** ±2.31 (-64.2)
CD-1-0.01	**-60.87** ±1.31 (-60.9)	**-61.31** ±1.55 (-61.3)	**-61.31** ±1.56 (-61.3)
PCD-1-0.1	**-58.80** ±3.03 (-199.4)	**-59.08** ±1.53 (-94.5)	**-58.53** ±2.42 (-177.3)
PCD-1-0.05	**-57.52** ±2.07 (-101.5)	**-56.97** ±1.74 (-64.3)	**-56.87** ±2.27 (-83.3)
PCD-1-0.01	**-57.31** ±1.34 (-57.3)	-56.88 ±0.75 (-56.9)	**-56.65** ±0.76 (-56.6)
PT$_{10}$-0.1	**-52.79** ±1.92 (-53.0)	**-51.99** ±1.29 (-52.5)	**-51.88** ±1.02 (-52.5)
PT$_{10}$-0.05	**-52.80** ±1.62 (-52.9)	**-52.34** ±1.15 (-52.5)	**-52.25** ±1.06 (-52.4)
PT$_{10}$-0.01	**-57.23** ±1.34 (-57.2)	**-56.76** ±0.77 (-56.8)	**-56.69** ±0.77 (-56.7)
Flipped Shifting Bar			
CD-1-0.2	**-20.79** ±1.11 (-20.8)	**-20.70** ±1.01 (-20.7)	**-20.64** ±1.05 (-20.7)
CD-1-0.1	**-21.07** ±0.91 (-21.1)	**-20.84** ±0.80 (-20.8)	**-21.01** ±0.87 (-21.0)
CD-1-0.05	**-22.52** ±0.61 (-22.5)	**-22.64** ±0.67 (-22.6)	**-22.61** ±0.68 (-22.6)
PCD-1-0.2	**-22.29** ±0.62 (-38.3)	**-22.21** ±0.85 (-34.0)	**-22.44** ±0.71 (-37.8)
PCD-1-0.1	**-21.88** ±0.65 (-23.9)	**-21.66** ±0.73 (-22.6)	**-21.75** ±0.59 (-23.8)
PCD-1-0.05	**-22.50** ±0.59 (-22.5)	**-22.36** ±0.60 (-22.4)	**-22.36** ±0.63 (-22.4)
PT$_{10}$-0.2	**-20.57** ±0.56 (-20.7)	**-20.47** ±0.57 (-20.6)	**-20.53** ±0.58 (-20.7)
PT$_{10}$-0.1	**-20.61** ±0.64 (-20.7)	**-20.62** ±0.61 (-20.7)	**-20.62** ±0.60 (-20.7)
PT$_{10}$-0.05	**-22.37** ±0.65 (-22.4)	**-22.48** ±0.72 (-22.5)	**-22.34** ±0.66 (-22.3)
MNIST			
CD-1-0.1	**-152.81** ±2.34 (-155.5)	**-153.16** ±2.57 (-154.9)	**-153.13** ±2.32 (-155.2)
CD-1-0.05	**-152.01** ±2.04 (-153.2)	**-152.34** ±3.61 (-153.1)	**-152.37** ±3.50 (-153.2)
CD-1-0.01	-153.20 ±1.89 (-153.2)	**-152.23** ±2.03 (-152.2)	**-152.13** ±2.04 (-152.1)
PCD-1-0.1	**-144.07** ±1.54 (-144.5)	**-144.12** ±2.12 (-145.2)	**-143.87** ±1.79 (-145.0)
PCD-1-0.05	**-142.19** ±1.62 (-143.2)	**-141.88** ±1.28 (-142.2)	**-141.98** ±1.96 (-143.2)
PCD-1-0.01	**-140.58** ±0.91 (-140.7)	**-140.40** ±0.66 (-140.7)	**-140.52** ±0.55 (-140.7)
PT$_{10}$-0.01	-143.68 ±1.53 (-153.0)	**-141.95** ±1.22 (-142.5)	-143.45 ±0.76 (-145.8)

Table 7: Maximum average LL on the training data for RBMs with 4 hidden units on (top) *Bars & Stripes*, and (middle) *Flipped Shifting Bar*, and (bottom) RBMs with 16 hidden units on *MNIST* when using an exponentially moving average with a sliding factor of 0.01.

if the training data and thus $\langle \mathbf{x} \rangle_d$ is known in advance the visible offsets should be fixed to this value independent of whether batch or mini-batch learning is used. However, the use of an exponentially moving average for approximating $\langle \mathbf{x} \rangle_d$ is reasonable if the training data is not known in advance, as well as for the approximation of the mean of the hidden representation $\langle \mathbf{h} \rangle_d$. We therefore used an exponentially moving average for the following experiments on the big data sets.

6.6 Control Experiments: Other Choices for the Offsets

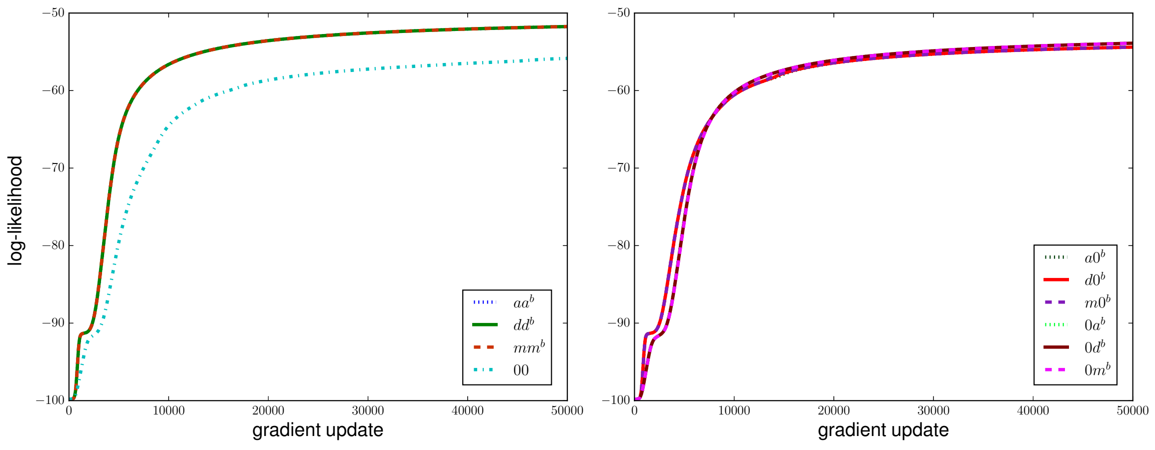
As discussed in Section 3, any offset value between 0 and 1 guarantees the flip invariance property as long as it is flipped simultaneously with the data. An intuitive and constant choice is to set the offsets to 0.5, which has also been proposed by Ollivier et al. (2011) and results in a symmetric variant 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*, we found that the performance is always worse compared to that of a normal RBM on the version of the data set that has fewer 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 3.1, we mentioned that using either $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_m$ or $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_m$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_d$ as offsets both lead to the same updates for the weights as the enhanced gradient. Using $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_m$ seems reasonable since the data mean is usually known in advance. As mentioned 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 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 as aa^b when PT_c is used for sampling, as shown in Figure 7(b).

We further tried variants like mm, $m0$, $0d$, $m0$, etc., but did not find better performance than that of dd. The variants that subtract an offset from both, visible and hidden variables outperformed or achieved the same performance as the variants that only subtract an offset from either visible or hidden variables. When the model expectation was used without an exponentially moving average either for $\boldsymbol{\mu}$ or $\boldsymbol{\lambda}$, or for both offsets we always observed the divergence when PT_c was used for sampling (results not shown).

Interestingly, if the exact gradient and offsets are used for training, no significant difference can be observed in terms of the LL evolution whether data mean, model mean or the average of both is used for the offsets, as shown in Figure 8. But centering both visible and hidden units still leads to better results than centering only one. Furthermore, the results illustrate that centered RBMs outperform normal binary RBMs also if the exact gradient is used for training both models. This further emphasizes that the worse performance of normal binary RBMs is caused by the properties of the gradient rather than by the gradient approximation.

The control experiments confirm our findings that centering both visible and hidden units is important and that the performance of dd, aa, and mm become similar when an exponentially moving average is used.



(a) Visible and hidden units centered

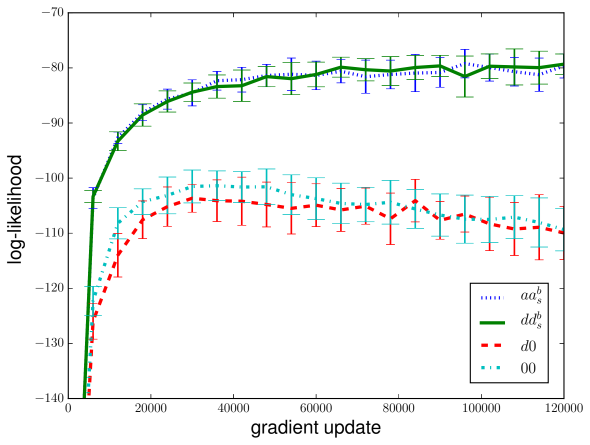
(b) Visible or hidden units centered

Figure 8: Evolution of the average LL on the training data for RBMs with 4 hidden units on the *Bars & Stripes* data set for the various centering methods when the exact gradient is used with the exact offsets and a learning rate of $\eta = 0.05$. The plots for aa^b, dd^b and mm^b overlay each other and so do $a0^b$, $d0^b$, $m0^b$, and also $0a^b$, $0d^b$ and $0m^b$. In (b) the plots for $a0^b$, $d0^b$, $m0^b$ can be distinguished from the plots for $a0^b$, $d0^b$, $m0^b$ in that they are initially slower and later lower.

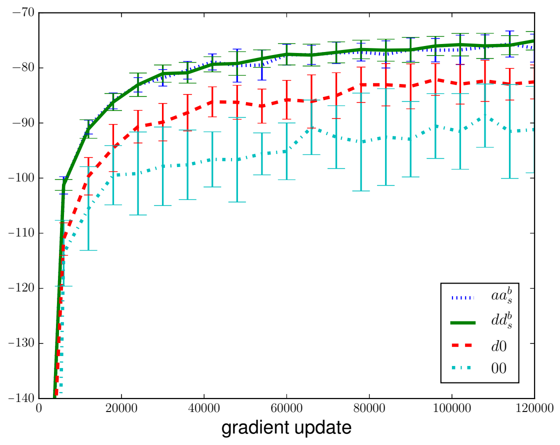
6.7 Experiments with Big RBMs

For the experiments in Section 6.1-6.6 we trained small models in order to be able to run many experiments and to evaluate the LL exactly. In this section we want to show that the results we observed on the toy problems and *MNIST* with RBMs having 16 hidden units carry over to more realistic settings. Furthermore, we want to investigate the generalization performance of the different models. In a first set of experiments we therefore trained the models *00*, *d0*, dd^b_s, and aa^b_s with 500 hidden units on *MNIST* and *Caltech*. The weight matrices were initialized with random values sampled from a Gaussian with zero mean and a standard deviation of 0.01, and visible and hidden biases and offsets were initialized as described in Section 4.2. The LL was estimated using Annealed Importance Sampling (AIS), where we used the same setup as described in the analysis of Salakhutdinov and Murray (2008).

Figure 9 shows the evolution of the average LL on the test data of *MNIST* over 25 trials for PCD-1 and PT_{20} for different centering versions. The models were trained for 200 epochs, each consisting of 600 gradient updates with a batch size of 100 and the LL was estimated every 10th epoch using AIS. Both variants dd^b_s and aa^b_s reach significantly higher LL values than *00* and *d0*. Also the standard deviation over the 25 trials indicated by the error bars is smaller for dd^b_s and aa^b_s than for *00* and *d0*, especially when PT_{20} is used for sampling. Furthermore, *00* and *d0* show divergence already after 30000 gradient updates when PCD-1 is used, while no divergence can be observed for dd^b_s and aa^b_s up to 120000



(a) PCD-1



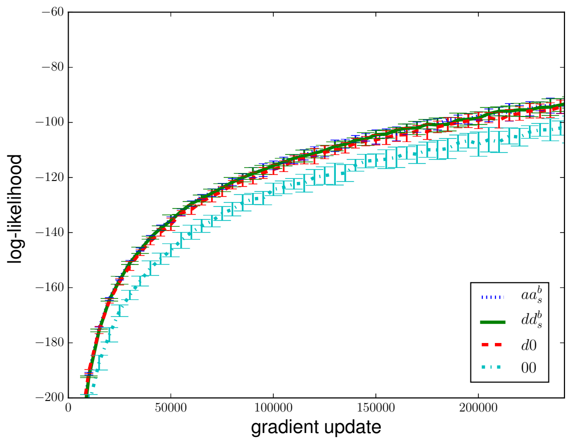
(b) PT_{20}

Figure 9: Evolution of the average LL on the test data of *MNIST* during training of different centering variants with 500 hidden units, using a learning rate of $\eta = 0.01$, and a sliding factor of 0.01. (a) When using PCD-1 and (b) when using PT_{20} for training. The error bars indicate the standard deviation of the LL over the 25 trials.

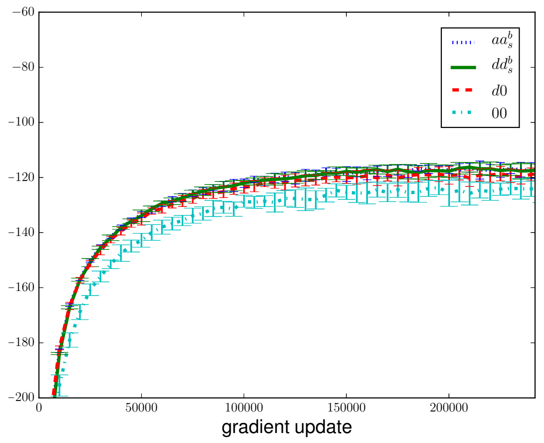
gradient updates. The evolution of the LL on the training data is not shown, since it is almost equivalent to the evolution on the test data. The superiority of centering over normal RBMs can also be observed for the other variants of *MNIST* as shown in Appendix D.

Figure 10 shows the evolution of the average LL on training and test data of the *Caltech* data set over 25 trials for different centering versions using PCD-1 with a batch size of 100 and either a learning rate of 0.001 or 0.01. The LL was estimated every 5000th gradient update using AIS. The results show that dd_s^b, aa_s^b and *d0* reach higher LL values than *00* for both learning rates and on training and test data. While dd_s^b and aa_s^b perform only slightly better than *d0* when a small learning rate is used, the difference becomes more prominent for a big learning rate. Figure 10(c) and (d) show that all models overfit to the training data. Nevertheless, dd_s^b and aa_s^b reach higher LL values on the test data and thus lead to a better generalization. The final average LL on the test data for dd_s^b and aa_s^b shown in Figure 10(b) is around -118. This is consistent with the reported performance of -120.0 for a normal RBM with 500 hidden units trained with PCD-1 (Marlin et al., 2010) and -114.75 when PT sampling is used in combination with the enhanced gradient (Cho et al., 2013b).

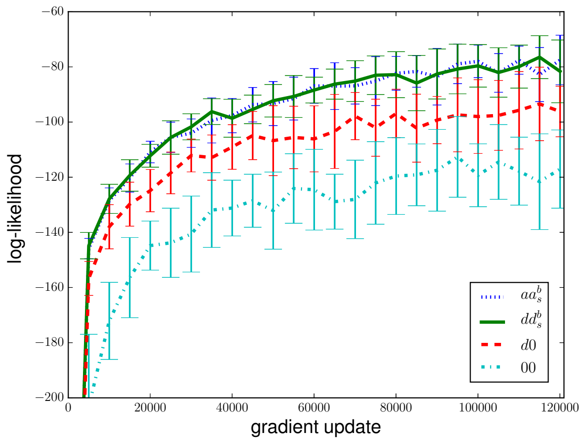
In a second set of experiments we extended our analysis to the eight *UCI binary* data sets used by Larochelle et al. (2010). The four different models *00*, *d0*, dd_s^b, and aa_s^b were trained with the same setup as before using PCD-1, a learning rate of 0.01, a batch size of 100 and a total number of 5000 epochs. All experiments were repeated 25 times and we trained either RBMs with 16 hidden units and calculated the LL exactly or RBMs with 200 hidden units using AIS for estimating the LL. Additionally we trained RBMs with 200 hidden variables with a smaller learning rate of 0.001 for 30000 epochs. Due to the long training time these experiments were repeated only 10 times. The maximum average LL for the test data is shown in Table 8. On seven out of eight data sets dd_s^b or aa_s^b reached the best result independent of whether 16 or 200 hidden units, or a learning rate of 0.01 or 0.001 were used. Whenever aa_s^b reached the highest value it was not significantly different from that of dd_s^b. Note that when training RBMs with 16 hidden units $d0$ reached comparable results to



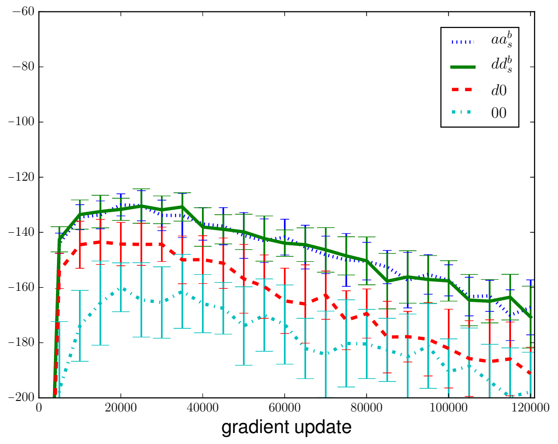
(a) Training data- learning rate 0.001



(b) Test data - learning rate 0.001



(c) Training data - learning rate 0.01



(d) Test data - learning rate 0.01

Figure 10: Evolution of the average LL on *Caltech* data set with different centering variants with 500 hidden units. The results on training and test data for a learning rate of $\eta = 0.001$ are shown in sub-figures (a) and (b), respectively, and for a learning rate of $\eta = 0.01$ in sub-figures (c) and (d), respectively. In both cases a sliding factor of 0.01 and PCD-1 was used. The error bars indicate the standard deviation of the LL over the 25 trials.

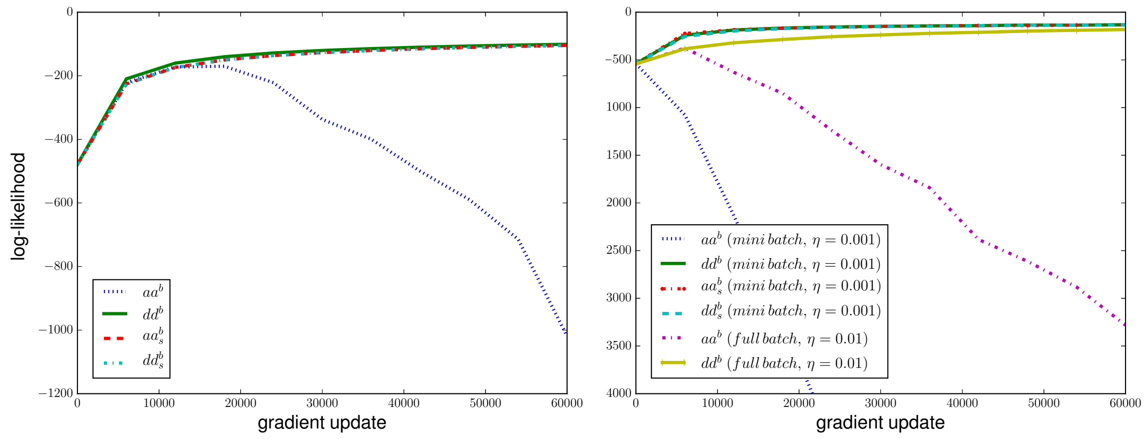
Data set	aa_s^b	dd_s^b	*d0*	*00*
16 hidden units - learning rate 0.01				
Adult	-18.09 ±0.62 (-18.18)	**-17.70** ±0.25 (-17.74)	-17.90 ±0.21 (-17.97)	-17.94 ±0.22 (-17.98)
CONNECT-4	-20.07 ±0.19 (-20.52)	**-19.89** ±0.21 (-19.90)	-20.14 ±0.24 (-20.16)	-20.59 ±0.29 (-20.74)
DNA	**-96.97** ±0.05 (-97.26)	**-96.97** ±0.04 (-97.25)	-97.01 ±0.04 (-97.21)	-97.03 ±0.05 (-97.19)
Mushroom	-16.83 ±0.57 (-17.11)	**-16.53** ±0.64 (-16.53)	**-16.76** ±0.56 (-17.11)	-17.05 ±0.59 (-17.44)
NIPS	**-276.37** ±0.16 (-279.13)	**-276.38** ±0.16 (-279.11)	-276.53 ±0.21 (-278.97)	-278.04 ±0.27 (-279.95)
OCR	**-45.81** ±0.13 (-45.81)	**-45.82** ±0.12 (-45.83)	**-45.84** ±0.12 (-45.86)	-45.97 ±0.17 (-45.99)
RCV1	-49.57 ±0.04 (-49.57)	-49.58 ±0.05 (-49.58)	-49.59 ±0.05 (-49.59)	**-49.53** ±0.04 (-49.53)
Web	**-29.99** ±0.05 (-29.99)	**-29.98** ±0.05 (-29.98)	**-30.20** ±0.70 (-30.86)	-30.75 ±0.11 (-38.82)
200 hidden units - learning rate 0.01				
Adult	-15.98 ±0.37 (-17.80)	**-15.55** ±0.25 (-16.22)	-16.65 ±0.61 (-18.96)	-16.91 ±0.78 (-19.24)
CONNECT-4	-14.85 ±0.24 (-23.31)	**-14.70** ±0.20 (-17.71)	-16.14 ±0.50 (-23.14)	-17.88 ±0.78 (-30.41)
DNA	-90.15 ±0.09 (-95.17)	**-90.12** ±0.10 (-94.13)	-90.78 ±0.10 (-95.93)	-91.13 ±0.12 (-97.62)
Mushroom	**-15.45** ±1.35 (-20.11)	**-15.61** ±1.23 (-19.93)	-16.28 ±0.88 (-21.59)	-16.42 ±1.61 (-21.90)
NIPS	**-270.81** ±0.05 (-291.82)	**-270.81** ±0.07 (-291.77)	-271.38 ±0.28 (-294.28)	-272.88 ±0.44 (-290.02)
OCR	**-29.75** ±0.50 (-30.18)	**-29.53** ±0.52 (-29.97)	-30.66 ±0.65 (-30.66)	-30.08 ±0.52 (-30.25)
RCV1	-47.13 ±0.12 (-47.13)	-47.14 ±0.12 (-47.16)	-47.14 ±0.10 (-47.19)	**-46.70** ±0.11 (-46.72)
Web	**-28.27** ±0.19 (-28.47)	**-28.27** ±0.19 (-28.58)	-28.50 ±0.51 (-29.16)	-28.35 ±0.14 (-28.71)
200 hidden units - learning rate 0.001				
Adult	-15.51 ±0.17 (-17.14)	$\underline{-14.90}$ ±0.07 (-14.90)	-15.27 ±0.22 (-15.48)	-15.16 ±0.21 (-15.39)
CONNECT-4	-13.73 ±0.17 (-15.95)	$\underline{-13.13}$ ±0.10 (-13.25)	-13.86 ±0.29 (-14.79)	-14.47 ±0.38 (-16.04)
DNA	-90.17 ±0.07 (-90.17)	$\underline{-90.15}$ ±0.07 (-90.15)	-90.45 ±0.09 (-90.45)	-90.73 ±0.13 (-90.73)
Mushroom	-13.44 ±0.75 (-13.96)	$\underline{-13.17}$ ±1.15 (-13.94)	-13.47 ±0.73 (-14.51)	-13.68 ±0.77 (-14.77)
NIPS	-270.71 ±0.03 (-313.30)	$\underline{-270.70}$ ±0.03 (-313.16)	-270.84 ±0.04 (-315.96)	-271.91 ±0.12 (-309.03)
OCR	-27.92 ±0.34 (-27.92)	$\underline{-27.70}$ ±0.15 (-27.75)	-28.24 ±0.21 (-28.34)	-28.04 ±0.39 (-28.29)
RCV1	-47.09 ±0.05 (-47.09)	-47.10 ±0.03 (-47.11)	-46.93 ±0.07 (-46.93)	$\underline{-46.56}$ ±0.07 (-46.56)
Web	-28.13 ±0.02 (-28.17)	$\underline{-28.11}$ ±0.06 (-28.11)	-28.16 ±0.05 (-28.20)	-28.13 ±0.05 (-28.15)

Table 8: Maximum average LL for the test data of the eight *UCI binary* data sets during training RBMs with (top) 16 hidden units and a learning rate of 0.01, (middle) 200 hidden units and a learning rate of 0.01 and (bottom) 200 hidden units and learning rate of 0.001 (since 10 trials are are not enough to perform a statistical significance test we simply underlined the best result). All models were trained using PCD-1 with a batch size of 100.

Data set	aa^b_s	dd^b_s	*d0*	*00*
16 hidden units - learning rate 0.01				
Adult	-17.93 ±0.61 (-18.01)	**-17.54** ±0.25 (-17.58)	-17.73 ±0.20 (-17.80)	-17.78 ±0.23 (-17.81)
CONNECT-4	-19.94 ±0.19 (-20.38)	**-19.75** ±0.25 (-19.77)	-20.01 ±0.24 (-20.03)	-20.46 ±0.28 (-20.61)
DNA	**-94.33** ±0.05 (-94.33)	**-94.33** ±0.06 (-94.33)	-94.44 ±0.04 (-94.44)	-94.42 ±0.05 (-94.42)
Mushroom	-16.55 ±0.56 (-16.82)	**-16.25** ±0.64 (-16.25)	**-16.46** ±0.56 (-16.80)	-16.75 ±0.60 (-17.15)
NIPS	**-255.03** ±0.25 (-255.05)	**-255.02** ±0.23 (-255.04)	-255.88 ±0.28 (-255.88)	-258.57 ±0.29 (-258.57)
OCR	**-45.88** ±0.12 (-45.88)	**-45.90** ±0.11 (-45.90)	**-45.90** ±0.12 (-45.93)	-46.03 ±0.16 (-46.05)
RCV1	-49.42 ±0.04 (-49.42)	-49.43 ±0.05 (-49.43)	-49.44 ±0.05 (-49.44)	**-49.38** ±0.04 (-49.38)
Web	**-29.79** ±0.05 (-29.79)	**-29.78** ±0.05 (-29.78)	**-29.97** ±0.70 (-30.62)	-30.64 ±0.11 (-38.61)
200 hidden units - learning rate 0.01				
Adult	-15.19 ±0.62 (-16.29)	**-14.51** ±0.43 (-14.80)	-15.83 ±0.61 (-17.17)	-16.00 ±0.79 (-17.51)
CONNECT-4	-14.39 ±0.24 (-22.50)	**-14.22** ±0.20 (-17.02)	-15.74 ±0.49 (-22.32)	-17.58 ±0.76 (-29.55)
DNA	**-60.19** ±1.20 (-60.69)	**-59.75** ±1.45 (-59.75)	**-60.31** ±1.64 (-60.31)	-61.11 ±1.54 (-61.71)
Mushroom	**-14.90** ±1.34 (-19.14)	**-15.06** ±2.09 (-19.00)	**-15.70** ±2.88 (-20.52)	-15.87 ±1.63 (-20.87)
NIPS	-180.48 ±0.24 (-180.48)	-180.51 ±0.31 (-180.51)	**-180.29** ±0.54 (-180.29)	-184.84 ±0.62 (-184.84)
OCR	**-28.97** ±0.49 (-29.39)	**-28.76** ±0.52 (-29.20)	-29.84 ±0.63 (-29.84)	-29.34 ±0.51 (-29.49)
RCV1	-45.83 ±0.12 (-45.83)	-45.85 ±0.11 (-45.87)	-45.92 ±0.18 (-45.97)	**-45.60** ±0.11 (-45.61)
Web	**-26.22** ±0.67 (-26.22)	-26.28 ±0.20 (-26.32)	-26.57 ±0.66 (-26.75)	-26.34 ±0.38 (-26.39)
200 hidden units - learning rate 0.001				
Adult	-14.61 ±0.52 (-15.96)	$\underline{-13.85}$ ±0.06 (-13.85)	-14.07 ±0.22 (-14.17)	-14.03 ±0.21 (-14.11)
CONNECT-4	-13.15 ±0.17 (-15.20)	$\underline{-12.54}$ ±0.11 (-12.62)	-13.28 ±0.29 (-14.10)	-13.98 ±0.38 (-15.39)
DNA	-70.22 ±0.08 (-70.22)	-70.24 ±0.08 (-70.24)	-69.66 ±0.06 (-69.66)	$\underline{-69.40}$ ±0.08 (-69.40)
Mushroom	-12.76 ±0.75 (-13.21)	$\underline{-12.43}$ ±1.18 (-13.21)	-12.69 ±0.72 (-13.68)	-13.05 ±0.76 -13.92)
NIPS	-148.14 ±0.30 (-148.14)	-148.30 ±0.31 (-148.30)	$\underline{-147.33}$ ±0.30 (-147.33)	-150.91 ±0.30 (-150.91)
OCR	-27.06 ±0.33 (-27.06)	$\underline{-26.90}$ ±0.14 (-26.95)	-27.41 ±0.20 (-27.52)	-27.28 ±0.39 (-27.53)
RCV1	-45.83 ±0.04 (-45.83)	-45.85 ±0.05 (-45.85)	-45.73 ±0.07 (-45.73)	$\underline{-45.47}$ ±0.06 (-45.47)
Web	-26.44 ±0.06 (-26.45)	-26.36 ±0.07 (-26.36)	$\underline{-26.32}$ ±0.05 (-26.33)	-26.35 ±0.10 (-26.35)

Table 9: Maximum average LL for the training data on the eight *UCI binary* data sets during training RBMs, with (top) 16 hidden units and a learning rate of 0.01, (middle) 200 hidden units and a learning rate of 0.01 and (bottom) 200 hidden units and a learning rate of 0.001 (since 10 trials are are not enough to perform a statistical significance test we simply underlined the best result). All models were trained using PCD-1 with a batch size of 100

.



(a) MNIST LL on test data

(b) Caltech LL on test data

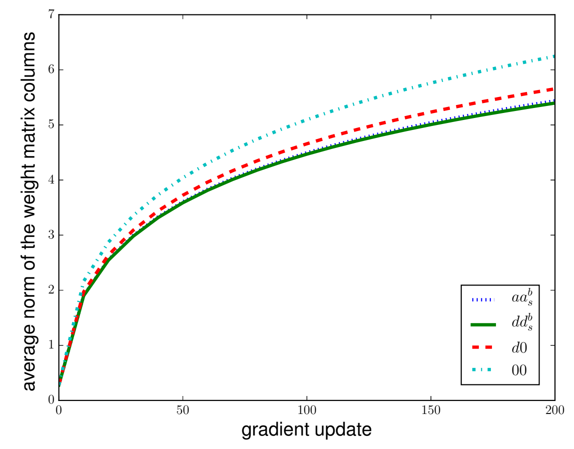
Figure 11: Evolution of the LL of exemplary trials for the different centering variants aa^b, dd^b, aa_s^b and dd_s^b on (a) *MNIST* and (b) *Caltech* during training using PT$_{20}$ with a batch size of 100, 500 hidden units and a learning rate of 0.001. For *Caltech* aa^b and dd^b were also trained in full batch mode with PT$_{20}$ and a learning rate of 0.01.

dd_s^b on some data sets. Only on the *RCV1* data set, 00 lead to better LL values than the centered RBMs for both 16 and 200 hidden units. It seems that the convergence rate on the *RCV1*, *OCR* and *WEB* data set is rather low for all models since the difference between the highest and the final LL values is rather small, indicating that no divergence or overfitting has happened so far. This can also be observed on the training data shown in Table 9. The *DNA* data set and the *NIPS* data set overfitted to the training data as indicated by the fact that the LL only decreased for the test data. In contrast, on the remaining three data sets *ADULT*, *CONNECT-4* and *MUSHROOM* the divergence can be observed on training and test data. Finally note that all eight data sets contain more zeros than ones in the current representation as mentioned in Section 5.1. Thus, the performance of the normal RBM would be even worse on the flipped data sets while for the centering variants it would stay the same (results not shown). Consistent with the experiments on small models, the results from nine of the ten real world data sets clearly support the superiority of centered over normal RBMs, show that centering visible and hidden units in RBMs is important for yielding good models and that averaged over all data sets dd_s^b performs better than aa_s^b.

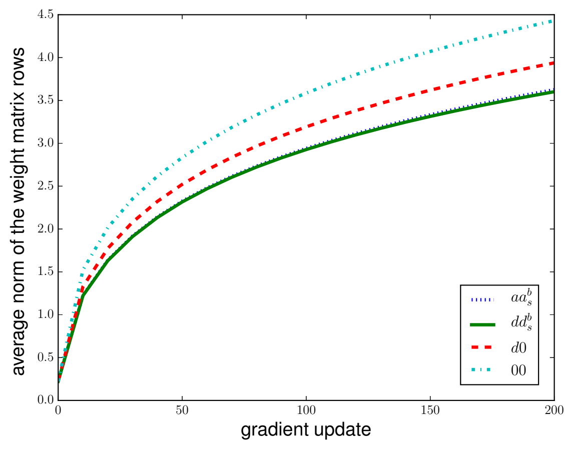
To emphasize that the divergence problems induced by using the model means as offsets also occurs for big models when no moving average is used, we trained RBMs with 500 hidden units on *MNIST* and *Caltech* using PT$_{20}$ with a learning rate of 0.001 and a batch size of 100. In addition, we trained aa^b and dd^b on *Caltech* using full batch learning and a learning rate of 0.01. Figure 11 shows that aa^b diverges while dd^b and the corresponding centering versions using a moving average, aa_s^b and dd_s^b, show no divergence. The divergence for aa^b even occurs in full batch training as shown in Figure 11(b).

6.8 Investigating the Effect of Centering on the Model Parameters

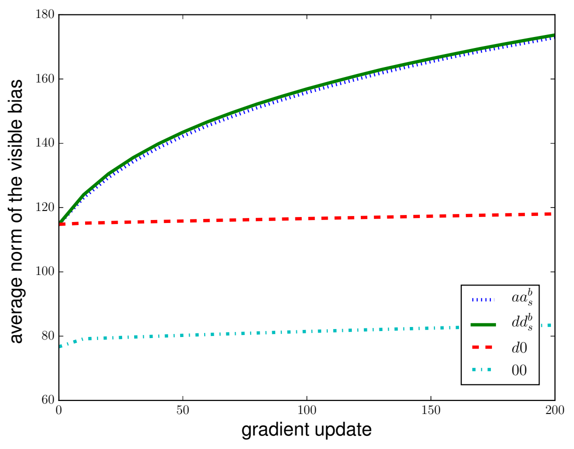
One explanation why centering works has already been given by Montavon and Müller (2012), who found that centering leads to a better conditioned optimization problem. Furthermore, Cho et al. (2011) have shown that when the enhanced gradient is used for training the update directions for the weights are less correlated than when the standard gradient is used, which allows to learn more meaningful features. From our analysis in Section 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



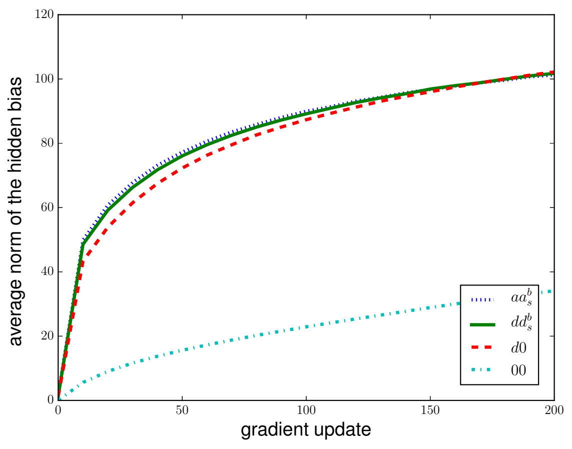
(a) Norm of the weight matrix colums



(b) Norm of the weight matrix rows



(c) Norm of the visible bias vector



(d) Norm of the hidden bias vector

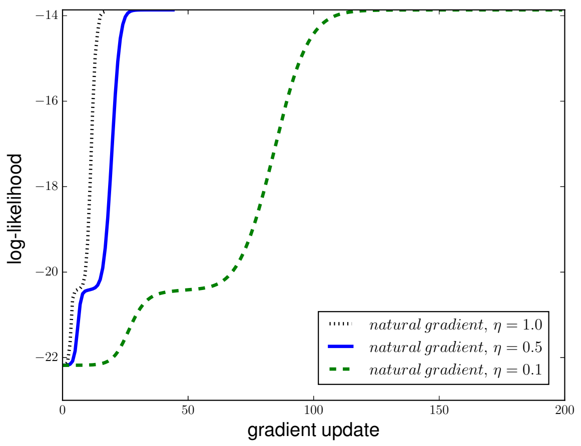
Figure 12: Evolution of the average Euclidean norm of the parameters of the RBMs with 500 hidden units trained on *MNIST*.

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 12, where it can be seen that the row and column norms (see Figure 12a and 12b) 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 (see Figure 12c and 12d) 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* and *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. This dependence between the mean of the hidden variables and the bias of the visible variables can also be seen from Equation (11) where the transformation of the visible bias depends on the offset of the hidden variables.

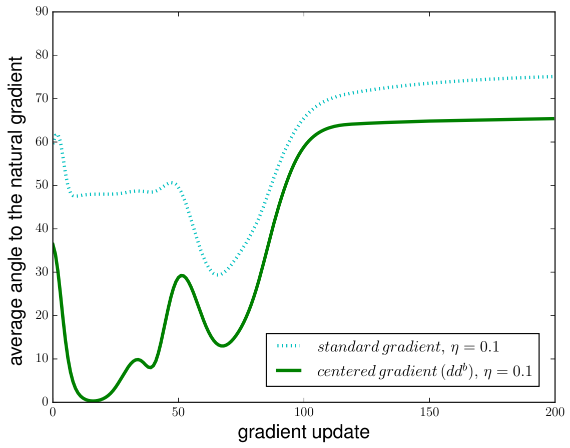
6.9 Comparision with the Natural Gradient

The results in Section 6.7 indicate that one explanation for the better performance of centering and thus the centered gradient compared to the standard gradient is the decoupling of the bias and weight parameters. As described in Section 2.2 the natural gradient is independent of the parameterization of the model distribution. Thus, it is also independent of how the mean information is stored in the parameters and should not suffer from the described bias-weight coupling problem. For the same reason it is also invariant under changes of the representation of the data distribution (for example variable flipping). 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.

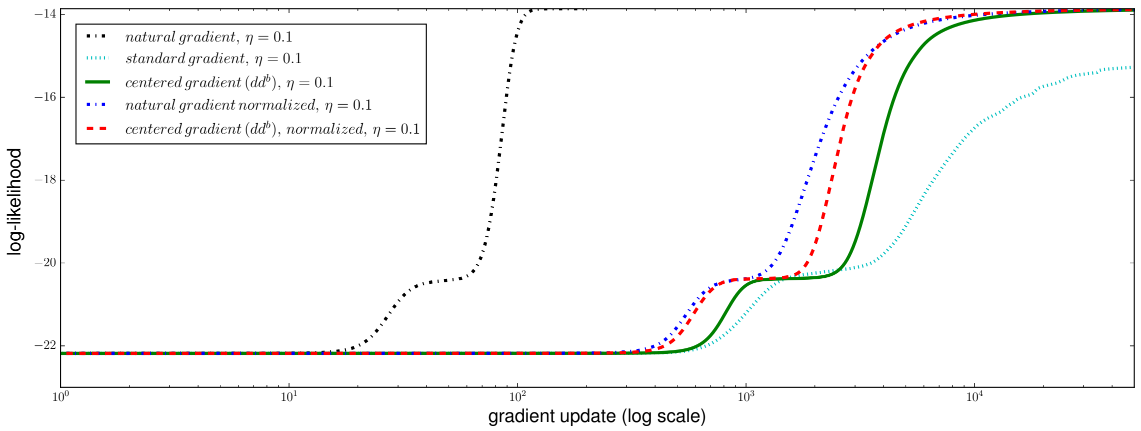
To verify this hypothesis empirically, we trained small RBMs with 4 visible and 4 hidden units using the exact natural gradient on the 2×2 *Bars & Stripes* data set. After each gradient update the different exact gradients were calculated and the angle between the centered and the natural gradient as well as the angle between the standard and the natural gradient were calculated. The results are shown in Figure 13 where Figure 13(a) shows the evolution of the average LL when the exact natural gradient is used for training with different learning rates. Figure 13(b) 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 results can also be observed for the *Shifting Bar* data set and when the standard or centered gradient is used for training (results not shown).



(a) LL evolution of the natural gradient



(b) Angle between standard natural gradient



(c) Comparision of the LL evolution of standard, natural and centered gradient

Figure 13: Comparison of the centered gradient, standard gradient, and natural gradient for RBMs with 4 visible and 4 hidden units trained on *Bars & Stripes 2×2*. (a) The average LL evolution on the training data over 25 trials when the natural gradient is used for training with different learning rates, (b) 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 (c) average LL evolution on the training data over 25 trials when either the natural gradient, standard gradient, or centered gradient is used for training.

Note how fast the natural gradient reached 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 quickly the natural gradient converges, we compared the average LL evolution for the standard, centered and natural gradient, as shown in Figure 13(c). Although much slower than the natural gradient, the centered gradient reached 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.

To verify that the better performance of natural and centered gradient is not only due to larger gradients resulting in bigger step sizes, we also analyzed the LL evolution for the natural and centered gradient when they are scaled to the norm of the standard gradient before updating the parameters. The results are shown in Figure 13(c). The natural gradient still outperforms the other methods but it becomes significantly slower than when used with its original norm. The reason why the norm of the natural gradient is somehow optimal can be explained by the fact that it ensures constant progress regardless of the curvature of the manifold of probability distributions as explained in (Desjardins et al., 2013). Like a Newton step the Fisher metric results in an automatic local step size adaption such that even a learning rate of 1.0 can be used as illustrated in Figure 13(a).

Interestingly, if the length of the natural gradient and the centered gradient are normalized to the length of the standard gradient and therefore the optimal step size is ignored, the centered gradient becomes almost as fast as the natural gradient. The fact that the normalization of the centered gradient increases the resulting learning speed shows that the norm of the centered gradient is smaller than the norm of the standard gradient. Therefore, the worse performance of the standard gradient does not result from the length but the direction of the gradient.

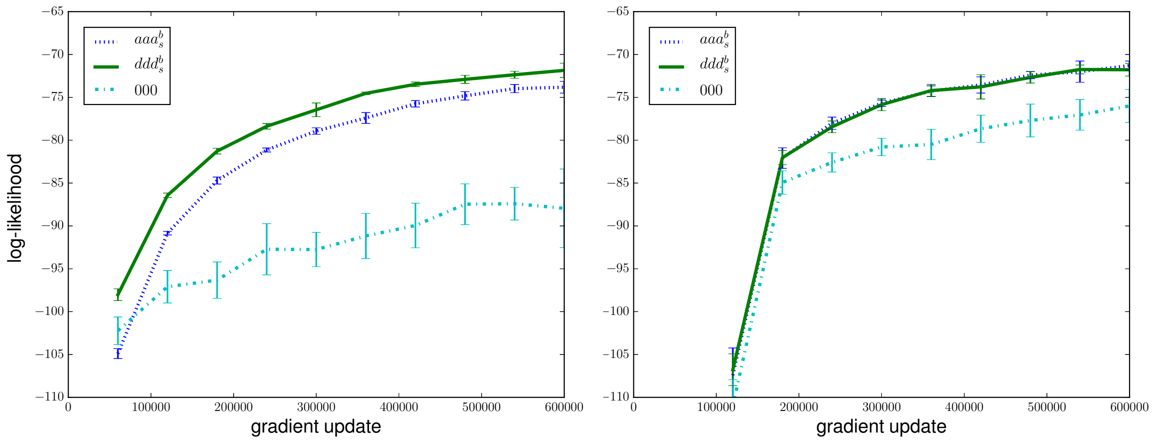
To conclude, these results support our assumption that the centered gradient is closer to the natural gradient and that it is therefore preferable over the standard gradient (Melchior et al., 2013; Fischer, 2014), which has recently also been confirmed by Grosse and Salakhudinov (2015).

6.10 Experiments with DBMs

When centering was first applied to DBMs by Montavon and Müller (2012) the authors have only seen an improvement via centering for locally connected DBMs. Due to our observations for RBMs and the structural similarity between RBMs and DBMs (a DBM is an RBM with restricted connections and partially unknown input data as discussed in Section 3.2) we expect that the observed benefit of centering also carries over to fully connected DBMs. To verify this assumption and empirically investigate the different centering variants for DBMs we performed extensive experiments on the big data sets listed in Section 5.1.

Training the models and evaluating the lower bound of the LL was performed as originally proposed for normal DBMs by Salakhutdinov and Hinton (2009). The authors have also proposed to pre-train DBMs in a layer-wise fashion based on RBMs (Hinton and Salakhutdinov, 2012). In our experiments we trained all models with and without pre-training to investigate the effect of pre-training in both normal and centered DBMs. For pre-training we used the same learning rate and the same offset type as in the final DBM models. Note that we keep using the term "average LL" although it is precisely speaking only the lower bound of the average LL, which has been shown to be rather tied (Salakhutdinov and Hinton, 2009). For the estimation of the partition function we again used AIS where we doubled the number of intermediate temperatures compared to the setting in the RBM experiments to 29.000. We continue using the shorthand notations introduced for RBMs also for DBMs with the only difference that we add a third letter to indicate the offset used in the second hidden layer, such that 000 corresponds to the normal binary DBM, and ddd_s^b and aaa_s^b correspond to the centered DBMs using the data mean and the average of data and model mean as offset, respectively. Due to the large number of experiments and the high computational cost (especially for estimating the LL) the experiments were repeated only 10 times and we focused our analysis on normal DBMs (000) and fully centered DBMs (ddd_s^b, aaa_s^b) only.

Again, we begin our analysis with the *MNIST* data set on which we trained normal and centered DBMs with 500 hidden units in the first and 500 units in the second hidden layer. Training was done using PCD-1 with a batch size of 100, a learning rate of 0.001 and in case of centering a sliding factor of 0.01 for the extensive amount of 1.000 epochs (600000



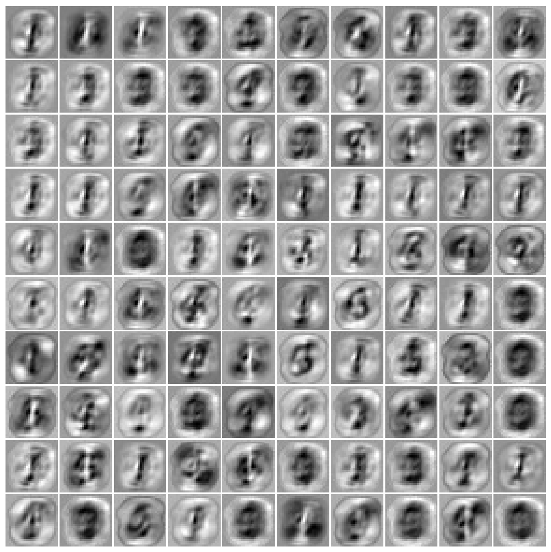
(a) Without pre-training

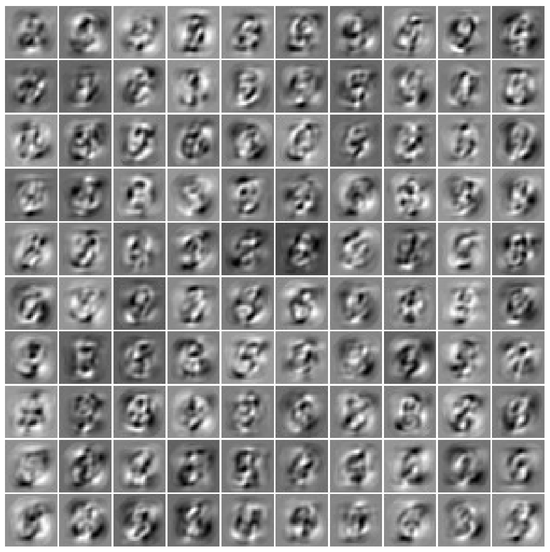
(b) With pre-training

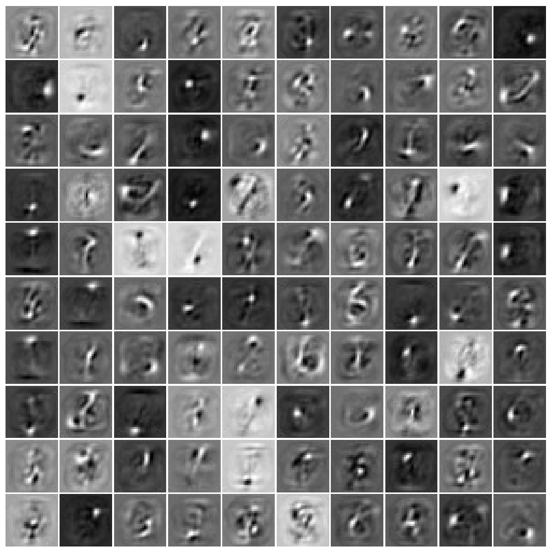
Figure 14: Evolution of the average LL on the test data of the *MNIST* data set for DBMs with 500 hidden units on the first and second layer. The different variants aaa_s^b, ddd_s^b and 000 were either trained (a) without pre-training or (b) with pre-training each DBM layer for 120000 gradient updates (200 epochs). In both cases PCD-1 with a learning rate of $\eta = 0.001$ and for centering a sliding factor of 0.01 was used. The error bars indicate the standard deviation of the average LL over the 10 trials. We skipped evaluating the initial model in (a) and started the LL evaluation in (b) after the 200 epochs of pre-training to roughly account for the computation overhead of pre-training

gradient updates). The evolution of the average LL on the test data without pre-training is shown in Figure 14(a), while the evolution of the average LL on the training data is not shown since it is almost equivalent. Both centered DBMs reach significantly higher LL values with a much smaller standard deviation between the trials than normal DBMs (as indicated by the error bars) and ddd_s^b performs slightly better than aaa_s^b. These findings are different to the observations of Montavon and Müller (2012) who reported an improvement of the model quality in terms of LL through centering only for locally connected DBMs. This might be due to the different training setup (for example a different learning rate, batch size, shorter training time, using ddd_s^l instead of ddd_s^b, or approximation of the data dependent part of the LL gradient by Gibbs sampling instead of optimizing the lower bound of the LL). Figure 14(b) shows the evolution of the average LL on the test data for the same models with pre-training for 120000 gradient updates (200 epochs). The evolution of the average LL on the training data was again almost equivalent and is thus not shown. It can be seen that ddd_s^b has approximately the same performance with and without pre-training but the performance of aaa_s^b improves by pre-training such that a similar LL level as for ddd_s^b is reached. Pre-training allows 000 to reach better LL than without pre-training, however it is still significantly worse compared to the centered DBMs with or without pre-training. By comparing these results with the results for RBMs with 500 hidden units trained on *MNIST* shown in Figure 9(a) we see that all DBMs reach higher LL values than the corresponding RBM models.

The higher layer representations in DBMs highly depend on the data driven lower layer representations. Therefore, we expect to see a qualitative difference between the second layer receptive fields or filters, given by the columns of the weight matrices in centered and normal DBMs. We did not visualize the filters of the first layer since all models showed the well known stroke like structure, which can be seen for RBMs in the review paper by Fischer and Igel (2014) for example. We visualized the receptive fields of the second layer by the linear back-projection of the second layer filters into the input space given by the matrix product of first and second layer weight matrix. The corresponding back projected second layer filters of 000 and ddd_s^b are shown Figure 15(a) and (b), respectively. It can be seen that many second layer filters of 000 are roughly the same and thus highly correlated. Moreover, they seem to represent some kind of mean information. Whereas the filters for ddd_s^b have much more diverse and less correlated structures than the filters of the normal DBM. When pre-training is used the filters of 000 become more diverse and the filters of both 000 and ddd_s^b become more selective as can be seen in Figure 15(c) and (d), respectively. The differences in selectivity of the filters between the different DBM variants can also be seen from the average mean field activation of the variables in the second hidden layer. As shown in Figure 16(a) without pre-training the average activation given the training data is approximately 0.5 for all hidden units of ddd_s^b while for aaa_s^b it is a bit less balanced and for 000 the units tend to be either active or inactive all the time. The results in Figure 16(b) illustrate that the average activities for all models become less balanced when pre-training is used, which also reflects the higher selectivity of the filters as shown in Figure 15(c) and (d). While the second layer hidden activities of ddd_s^b and aaa_s^b stay in a reasonable range, they become extremely selective for 000 where 300 out of 500 units are inactive most of the time. Therefore, the filters, the average activation and the evolution of the LL indicate that normal RBMs have difficulties in making use of the second hidden layer.


(a) 000 without pre-training


(b) ddd^b without pre-training


(c) 000 with pre-training

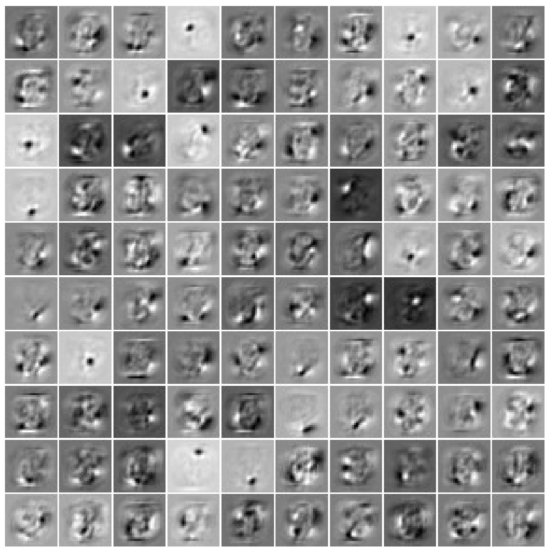
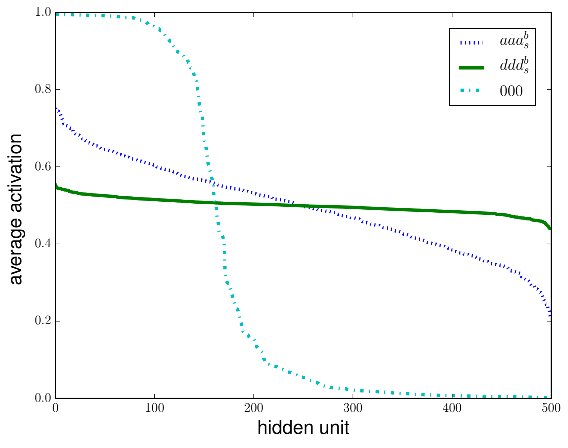
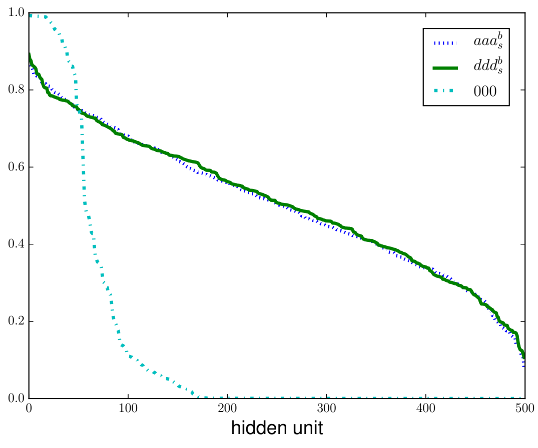

(d) ddd^b with pre-training

Figure 15: Random selection of 100 linearly projected filters of the second hidden layer for (a) 000 and (b) ddd^b without pre-training and (c) 000 and (d) ddd^b without 200 epochs pre-training. The filters have been normalized independently such that the structure can be seen better.



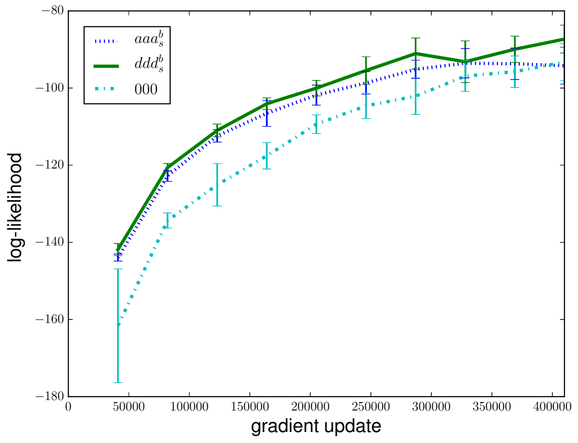
(a) Without pre-training

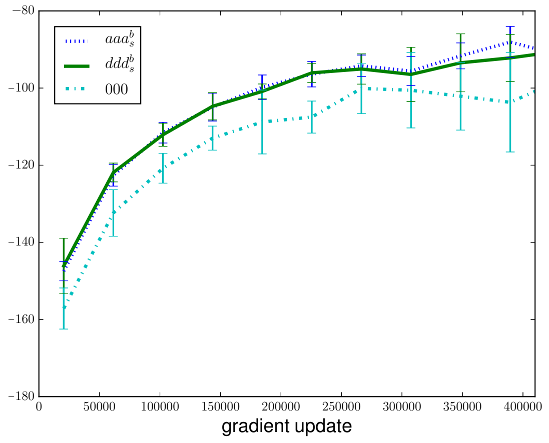


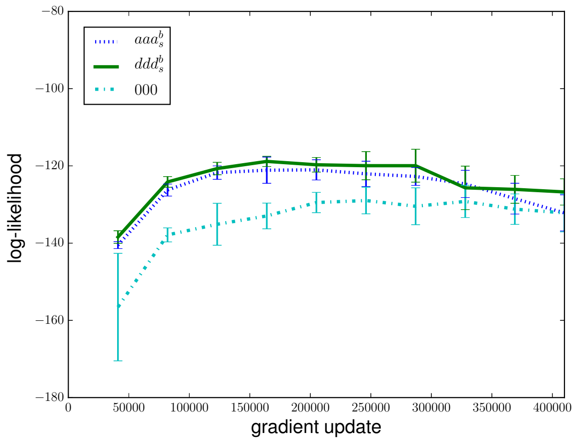
(b) With pre-training

Figure 16: Ordered average mean field activation of the 500 hidden units in the second layer of the different DBMs given the training data (a) without pre-training and (b) with pre-training. Sampling the states or using the conditional probabilities of the variables instead of taking the mean field activation leads to almost equivalent results.

We continue our analysis with experiments on the *Caltech* data set on which we again trained normal and centered DBMs with 500 hidden units on the first and 500 hidden units on the second hidden layer. Training was also done using PCD-1 with a batch size of 100, a learning rate of 0.001 and in case of centering a sliding factor of 0.01. Since the training data has only 41 batches the models were trained for the extensive amount of 10000 epochs (410000 gradient updates). Figure 17 shows the average LL on the test data (c) without and (d) with 500 epochs pre-training. In addition, Figure 17(a) and (b) show the corresponding average LL on the training data demonstrating that all models overfitted to the training data of the *Caltech* data set. The results are consistent with the results on *MNIST*. 000 performs worse than centering on training and test data independently of whether pre-training is used or not. Furthermore, aaa^b_s seems to perform slightly worse than ddd^b_s without pre-training, while the performance becomes equivalent if pre-training is used. But in contrast to the results for *MNIST*, on the *Caltech* data set all methods perform worse with pre-training. This negative effect of pre-training becomes even worse when the number of pre-training epochs is increased. In the case of 2.000 epochs of pre-training for example, ddd^b_s and aaa^b_s still perform better than 000 but the maximal average LL among all models, which was reached by ddd^b_s was only -98.1 for the training data and -141.4 for the test data, compared to -90.4 and -124.0 when 500 epochs of pre-training were used, and -87.3 and -118.8 when no pre-training was used. Without pre-training the LL values are comparable to the results when an RBM with 500 hidden units is trained on *Caltech* as shown in Figure 10, illustrating that in terms of the LL a DBM does not necessarily perform better than an RBM. We also examined the filters and the average hidden activities for the


(a) LL on the training data without pre-training


(b) LL on the training data with pre-training


(c) LL on the test data without pre-training

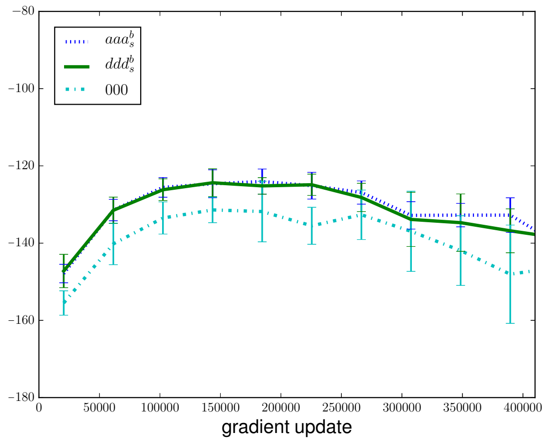

(d) LL on the test data with pre-training

Figure 17: Evolution of the average LL on the *Caltech* data set for different DBMs (aaa^b_s, ddd^b_s, and 000) with 500 units in the first and 500 units in second hidden layer. LL on (a) the training data and (c) the test data without pre-training and the LL on (b) the training data and (d) the test data with 500 epochs (20500 gradients updates) of pre-training. The models were trained using PCD-1 with a batch size of 100, a sliding factor of 0.01 and a learning rate of $\eta = 0.001$. The error bars indicate the standard deviation of the LL over the 10 trials. We skipped evaluating the initial model and (a) and (c) start after the 500 epochs of pre-training to roughly account for the computation overhead of pre-training.

training data of *Caltech*. Both led to the same conclusions as the results for *MNIST* and are therefore not shown here.

Finally, we also performed experiments on the eight *UCI binary* data sets described in Section 5.1 using the same training setup as for the corresponding RBM experiments. That is, the DBMs with 200 hidden units on the first and 200 hidden units on the second hidden layer were trained for 5.000 epochs with PCD-1, a batch size of 100, a learning rate of 0.01, and in the case of centering a sliding factor of 0.01. The LL was evaluated every 50th epoch and in the case of pre-training the models were pre-trained for 200 epochs. Table 10 shows the maximum average LL on the test data with and without pre-training. Without pre-training the results are consistent with the findings for RBMs. ddd^b performs better than 000 on all data sets except for RCV1 where 000 performs slightly better. The LL values for the DBMs are comparable but not necessarily better than the corresponding LL values for RBMs, which are shown in Table 8. In the case of the *WEB* data set for example the DBMs even perform worse than the RBMs . When pre-training is used the performance of all models, centered or normal, is worse than the performance of the corresponding DBMs without pre-training. For completeness Table 11 shows the maximum average LL on the training data leading to the same conclusions as the results on test data.

Data set	ddd^b_s			*000*		
No pre-training						
adult	$\underline{-15.54}$	±0.42	(-17.12)	-18.44	±1.15	(-24.92)
connect4	$\underline{-15.09}$	±0.39	(-40.83)	-18.15	±1.14	(-43.84)
dna	$\underline{-89.81}$	±0.13	(-92.57)	-91.18	±0.17	(-95.12)
mushrooms	$\underline{-15.24}$	±0.60	(-19.68)	-17.21	±1.16	(-27.71)
nips	$\underline{-270.35}$	±0.09	(-360.59)	-275.43	±1.81	(-360.56)
ocr letters	$\underline{-30.37}$	±0.39	(-32.23)	-31.56	±0.93	(-32.74)
rcv1	-46.83	±0.08	(-47.26)	$\underline{-46.51}$	±0.56	(-47.88)
web	$\underline{-30.02}$	±0.59	(-72.88)	-30.35	±1.19	(-79.36)
With pre-training						
adult	$\underline{-18.86}$	±2.74	(-21.43)	-21.64	±1.64	(-40.42)
connect4	$\underline{-27.38}$	±1.52	(-32.13)	-41.21	±4.07	(-52.04)
dna	$\underline{-89.87}$	±0.11	(-94.03)	-91.06	±0.19	(-97.48)
mushrooms	-24.23	±5.43	(-35.07)	$\underline{-21.42}$	±6.32	(-35.82)
nips	$\underline{-272.92}$	±0.16	(-404.11)	-276.88	±2.33	(-378.76)
ocr letters	-36.89	±1.44	(-39.76)	$\underline{-32.25}$	±1.18	(-35.01)
rcv1	-47.79	±0.84	(-49.30)	$\underline{-46.90}$	±0.36	(-48.36)
web	$\underline{-31.10}$	±0.14	(-81.93)	-32.43	±1.46	(-47.73)

Table 10: Maximum average LL on test data of the eight *UCI binary* data sets for DBMs with 200 units in the first and second hidden layer. For training (top) without pre-training and (bottom) with 200 epochs of pre-training. PCD-1 with a learning rate of 0.01 and a batch size of 100 was used. (The best result is underlined.)

DATA SET	ddd_s^b			000		
NO PRE-TRAINING						
ADULT	<u>-14.65</u>	±0.37	(-15.49)	-17.88	±1.16	(-23.04)
CONNECT4	<u>-14.68</u>	±0.38	(-39.74)	-17.82	±0.63	(-42.78)
DNA	<u>-62.00</u>	±1.11	(-62.42)	-62.48	±0.17	(-62.98)
MUSHROOMS	<u>-14.74</u>	±1.54	(-18.61)	-16.62	±1.13	(-26.53)
NIPS	<u>-107.09</u>	±2.11	(-107.09)	-114.28	±2.81	(-114.28)
OCR LETTERS	<u>-29.15</u>	±0.67	(-30.91)	-30.41	±0.90	(-31.57)
RCV1	<u>-45.75</u>	±0.07	(-46.18)	-45.80	±0.82	(-47.17)
WEB	<u>-29.37</u>	±0.59	(-69.70)	-29.68	±1.18	(-77.11)
WITH PRE-TRAINING						
ADULT	<u>-17.11</u>	±2.70	(-19.66)	-21.42	±1.63	(-38.21)
CONNECT4	<u>-26.62</u>	±6.66	(-31.14)	-40.53	±4.13	(-51.29)
DNA	<u>-59.97</u>	±0.49	(-60.37)	-61.16	±1.57	(-61.72)
MUSHROOMS	-23.89	±5.27	(-34.11)	<u>-20.59</u>	±6.33	(-34.86)
NIPS	<u>-114.51</u>	±3.76	(-118.29)	-118.90	±3.68	(-120.94)
OCR LETTERS	-35.39	±1.88	(-37.70)	<u>-30.90</u>	±1.14	(-33.21)
RCV1	-46.54	±0.87	(-48.07)	<u>-45.93</u>	±0.36	(-47.39)
WEB	<u>-30.41</u>	±0.15	(-78.45)	-31.68	±1.45	(-44.42)

Table 11: Maximum average LL on training data on the eight *UCI binary* data sets for DBMs with 200 hidden units on the first and second layer. For training (top) without pre-training and (bottom) with 200 epochs pre-training PCD-1, with a learning rate of 0.01, batch size of 100 was used. (The best result is underlined).

To summarize, the experiments described in this section show that centered DBMs reach higher LL values than normal DBMs. While pre-training leads to more selective filters in general, it is often even harmful for the model performance in terms of the LL.

6.11 Experiments with Auto Encoders

The benefit of centering in feed forward neural networks for supervised tasks has already been shown by Schraudolph (1998). In this section we analyze centering in a special kind of unsupervised feed forward neural networks, namely centered AEs as introduced in Section 4.1. We therefore trained normal and centered three layer AEs on the ten big data sets described in Section 5.1. To avoid trivial solutions we used tied weights and the number of output dimensions was 50 percent of the number of input/data dimensions. Since the data sets are binary we used the sigmoid non-linearity in encoder and decoder and the cross entropy cost function. Training was done using standard back propagation for 5000 epochs without any further modification. As for the RBM and DBM experiments, the weight matrices were initialized with random values sampled from a Gaussian with zero mean and standard deviation 0.01, and the biases and offsets were initialized as described in Section 4.2. The batch size was 100, the sliding factor 0.01 and we used a default learning rate

Data set - Learning rate	dd_s^b			00		
Test data						
mnist - 0.1	50.21472	±0.0256	(5000)	50.24859	±0.0200	(5000)
mnist - 0.5	50.01338	±0.0316	(5000)	56.36068	±0.7578	(22)
caltech - 0.01	44.38403	±0.2257	(2500)	49.21274	±0.2119	(1968)
caltech - 0.1	44.45882	±0.1620	(246)	48.59724	±0.3964	(206)
adult - 0.1	0.38837	±0.0229	(5000)	0.47460	±0.0181	(4676)
adult - 0.5	0.36825	±0.0220	(1526)	0.46086	±0.0155	(884)
connect4 - 0.1	0.03015	±0.0025	(5000)	0.03467	±0.0040	(5000)
connect4 - 0.5	0.02357	±0.0019	(1431)	0.02856	±0.0032	(1349)
dna - 0.01	34.34353	±0.1242	(2161)	34.77299	±0.1948	(2105)
dna - 0.1	34.35117	±0.1245	(216)	34.80547	±0.1823	(210)
mushrooms - 0.1	0.14355	±0.0117	(5000)	0.14226	±0.0081	(5000)
mushrooms - 0.5	0.08555	±0.0167	(3173)	0.09960	±0.0169	(2936)
nips - 0.01	183.21045	±0.6355	(2261)	188.65301	±0.7262	(2107)
nips - 0.1	183.31413	±0.6081	(226)	189.10982	±0.6674	(212)
ocr letters - 0.1	5.41182	±0.1994	(5000)	5.46969	±0.2138	(5000)
ocr letters - 0.5	4.91528	±0.1715	(3703)	5.30343	±0.2737	(1945)
rcv1 - 0.1	12.93456	±0.1562	(5000)	12.55443	±0.3097	(5000)
rcv1 - 0.5	12.32545	±0.3296	(4953)	12.16340	±1.2188	(2946)
web - 0.1	1.07535	±0.0174	(1425)	1.22756	±0.0121	(944)
Training data						
mnist - 0.1	50.03172	±0.0224	(5000)	50.06083	±0.0167	(5000)
mnist - 0.5	49.84428	±0.0253	(5000)	55.89110	±0.7516	(22)
caltech - 0.01	5.71437	±0.0235	(5000)	6.42052	±0.0392	(5000)
caltech - 0.1	0.57507	±0.0030	(5000)	0.62426	±0.0119	(5000)
adult - 0.1	0.04687	±0.0024	(5000)	0.03677	±0.0024	(5000)
adult - 0.5	0.03221	±0.0030	(1526)	0.04053	±0.0051	(894)
connect4 - 0.1	0.01465	±0.0007	(5000)	0.01187	±0.0006	(5000)
connect4 - 0.5	0.01022	±0.0002	(1431)	0.00914	±0.0004	(1349)
dna - 0.01	17.80686	±0.0812	(5000)	18.08306	±0.0665	(5000)
dna - 0.1	11.70726	±0.1018	(5000)	12.08309	±0.0995	(5000)
mushrooms - 0.1	0.06362	±0.0022	(5000)	0.05136	±0.0026	(5000)
mushrooms - 0.5	0.01995	±0.0007	(3173)	0.01768	±0.0013	(2936)
nips - 0.01	21.45218	±0.0614	(5000)	21.94275	±0.0509	(5000)
nips - 0.1	2.05141	±0.0067	(5000)	2.06286	±0.0054	(5000)
ocr letters - 0.1	4.97384	±0.1865	(5000)	5.00803	±0.2078	(5000)
ocr letters - 0.5	4.51704	±0.1642	(3703)	4.88637	±0.2677	(1945)
rcv1 - 0.1	11.95633	±0.1226	(5000)	11.62695	±0.2792	(5000)
rcv1 - 0.5	11.36861	±0.2974	(4953)	11.27664	±1.1136	(2947)
web - 0.1	0.06872	±0.0007	(5000)	0.05554	±0.0003	(5000)

Table 12: Average minimal reached cost value with standard deviation on test and training data of *MNIST*, *Caltech* and the eight*UCI binary* data sets for centered and normal three layer AEs with sigmoid units, cross entropy cost function, and the number of hidden units set to 50 percent of the number of input units (data dimensions). The average number of epochs needed to reach the minimal cost value is given in brackets, and (5000) indicates that convergence was not achieved during training.

of 0.1 for all experiments. Each experiment was repeated 10 times and we calculated the average minimal cost value, the corresponding standard deviation and the average number of epochs needed to reach the minimal cost value. A second set of experiments was performed with a learning rate of 0.5 when the average number of epochs needed to reach the minimal cost value on the test data was close or equal to 5000 epochs, or with a learning rate of 0.01 when the average number of epochs needed to reach the minimal cost value on the test data was less than 500 epochs. The results are given by Table 12, showing that (except for the RCV1 data set) centered AEs perform clearly better in terms of the average minimal cost value on the test data than normal AEs. On the training data normal AEs only perform slightly better on data sets where both models reached very small cost values anyway. We did not show the results for the validation sets since they are almost equivalent to the results for test data.

Interestingly, the result that centering only performs worse on the RCV1 data set is fully consistent with the findings for RBMs and DBMs. We inspected the RCV1 data set and its first and second order statistics but did not find anything conspicuous compared to the other data sets that might have explained why for this particular data set centering is not beneficial. However, learning is much slower for this data set when centering is used, which can also be seen by comparing the results for learning rate 0.1 an 0.5 in Table 12.

7. Conclusion

This work discussed centering in RBMs and DBMs, where centering is done by subtracting offset parameters from visible and hidden variables. Our theoretical analysis has yielded the following results:

1. Centered BMs/RBMs/DBMs and normal BMs/RBMs/DBMs are different parameterizations of the same model class (Section 3), such that any normal BM/RBM/DBM can be transformed to an equivalent centered BM/RBM/DBM an *vice versa*. This equivalence generalizes to ANNs in general, which justifies the use of centering in arbitrary ANNs (Section 4).

2. The LL gradient of centered binary BMs and thus of centered binary RBMs/DBMs is invariant under simultaneous flip of data and offsets, for any offset value in the range of zero to one. This leads to the desired invariance of the LL performance of the model under changes of data representation (Section 3 and Appendix A).

3. Training a centered BM/RBM/DBM can be reformulated as training a normal BM/RBM/DBM with a new parameter update, which we refer to as centered gradient (Section 3.1 and Appendix B).

4. From this new formulation follows that the enhanced gradient is 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 (Section 3.1 and Appendix C).

Our numerical analysis has yielded the following results:

1. Optimal performance of centered binary RBMs/DBMs is achieved when both, visible and hidden variables, are centered and the offsets are set to the expectations of the corresponding variable under data or model distribution (Section 6.1, 6.6 and 6.7).
2. Centered binary RBMs/DBMs reach significantly higher LL values than normal binary RBMs/DBMs (Section 6.7 and 6.10). As an example, centered binary RBMs with 500 hidden units achieved an average LL on the test data of *MNIST* around -75 compared to -88 for normal binary RBMs (Figure 9).
3. Initializing the bias parameters such that the RBM/DBM/AE is initially centered (that is $b_i = \sigma^{-1}(\langle x_i \rangle)$) can already improve the performance of a normal binary RBM. However, this initialization leads to a performance still worse than the performance of a centered binary RBM (Section 6.2) and is therefore no alternative to centering.
4. Using the model expectation (as for the enhanced gradient for example) can lead to a severe divergence of the LL when PCD or PT_c is used for sampling. This is caused by the correlation of offset and gradient approximation (Section 6.4).
5. The divergence can be prevented when an exponentially moving average for the approximations of the offset values is used (Section 6.5), which also stabilizes the training for other centering variants especially when the mini batch size is small.
6. Training centered binary RBMs/DBMs leads to smaller weight norms and larger bias norms compared to normal binary RBMs/DBMs. 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 (Section 6.9).
7. The direction of the centered gradient is closer to the natural gradient than that of the standard gradient and the natural gradient is extremely efficient for training RBMs if tractable (Section 6.9).
8. Centered binary DBMs reach higher LL values than normal binary DBMs independently of whether pre-training is used or not. Thus pre-training cannot be considered as an alternative to centering (Section 6.10).
9. While pre-training slightly helps normal binary DBMs on *MNIST* we did not observe an improvement through pre-training for centered binary DBMs. Furthermore, on all data sets other than *MNIST* pre-training led to lower LL values and the results became worse as longer pre-training was performed for normal and centered binary DBMs (Section 6.10).
10. The visual inspection of the learned filters, the average second hidden layer activities and the reached LL values suggest that normal binary DBMs have difficulties in making use of the third and higher layers (Section 6.10).
11. Centering also improves the performance in terms of the optimized loss for AEs, which supports our assumption that centering is beneficial not only for probabilistic models like RBMs and DBMs but also for ANNs in general (Section 6.11).

Based on our results we recommend to center all units in the network using the data mean and to use an exponentially moving average if the mini-batch size is rather small (< 100 for stochastic models and < 10 for deterministic models). Furthermore, we do not recommend to use pre-training for DBMs since it often deteriorates the results.

All results clearly support the superiority of centered RBMs/DBMs and AEs, which we believe will also extend to other models. Future work might focus on centering in BMs and also other probabilistic models such as the neural auto-regressive distribution estimator (Larochelle and Murray, 2011) or in recurrent neural networks such as long short-term memory (Hochreiter and Schmidhuber, 1997). An extension to also normalizing the variance of the units and a comparison to the recently proposed batch normalization (Ioffe and Szegedy, 2015) would also be of interest.

The implementation of the algorithms proposed and analyzed in this work are part of the Python library PyDeep publicly available at `https://github.com/MelJan/PyDeep`.

Acknowledgments

We would like to thank Nan Wang for helpful discussions, Tobias Glasmachers for his suggestions on the natural gradient part, and Alberto Escalante for his valuable corrections. We also like to thank the anonymous reviewers for helpful suggestions. Furthermore, Asja Fischer was 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).

Appendix A. Proof of Invariance for the Centered Gradient

In the following we show that the LL gradient of centered binary BMs and thus the LL gradient for centered binary RBMs and DBMs is invariant to flips of the variables if the corresponding offset parameters flip as well (see Melchior et al., 2013, for a proof specifically for RBMs). Since training a normal binary BM using the centered gradient (see Appendix B) is equivalent to training a centered binary BM, the proof also holds for the centered gradient.

In contrast to an RBM or DBM a BM is a fully connected graphical model and the energy for a centered BM and binary values $\mathbf{x} = (x_1, ..., x_N)$ is given by

$$E(\mathbf{x}) \;=\; -\sum_i (x_i - \mu_i)\, b_i - \sum_{i,j>i} (x_i - \mu_i)\, w_{ij}\, (x_j - \mu_j) \quad , \tag{18}$$

and the corresponding LL gradient w.r.t. the parameters w_{ij} and b_i is given by

$$\nabla w_{ij} \;=\; \langle (x_i - \mu_i)(x_j - \mu_j) \rangle_d - \langle (x_i - \mu_i)(x_j - \mu_j) \rangle_m \quad , \tag{19}$$

$$\nabla b_i \;=\; \langle (x_i - \mu_i) \rangle_d - \langle (x_i - \mu_i) \rangle_m = \langle x_i \rangle_d - \langle x_i \rangle_m \quad . \tag{20}$$

In the following $\boldsymbol{\theta}$ and $\nabla\boldsymbol{\theta}$ is used to jointly denote all parameters w_{ij} and b_i and their derivatives ∇w_{ij} and ∇b_i, respectively .

We begin by formalizing the invariance property for the energy.

Definition 1 *Let there be a binary random variable X_i. The variable X_i' is called 'flipped' or 'flip of X_i' if it takes the values $x_i' = 1 - x_i$ for any given state x_i of X_i.*

Definition 2 *Let there be a centered binary BM with parameters $\boldsymbol{\theta}$, offsets $\boldsymbol{\mu}$, binary random variables $\mathbf{X} = (X_1, \dots, X_N)$, and energy $E(\mathbf{x})$, and a second centered binary BM with parameters $\tilde{\boldsymbol{\theta}}$, offsets $\tilde{\boldsymbol{\mu}}$, binary random variables $\tilde{\mathbf{X}} = (\tilde{X}_1, \dots, \tilde{X}_N)$ and energy $\tilde{E}(\tilde{\mathbf{x}})$, where in the latter BM an arbitrary number of variables have been flipped. The energies $E(\mathbf{x})$ and $\tilde{E}(\tilde{\mathbf{x}})$ are called 'flip-invariant' or 'invariant to flips of variables' if $E(\mathbf{x}) = \tilde{E}(\tilde{\mathbf{x}})$ holds.*

Theorem 3 *Let the binary parameter f_i take the value -1 if the corresponding variable X_i, has been flipped and 1 otherwise. The energies $E(\mathbf{x})$ and $\tilde{E}(\tilde{\mathbf{x}})$ are 'flip-invariant' according to Definition 2 if the offsets flip simultaneously with the variables, compactly denoted by*

$$\tilde{x}_i = \frac{1-f_i}{2} + f_i x_i \ , \tag{21}$$

$$\tilde{\mu}_i = \frac{1-f_i}{2} + f_i \mu_i \ , \tag{22}$$

and if the parameters of the models are related in the following way

$$\tilde{w}_{ij} = f_i w_{ij} f_j \ , \tag{23}$$

$$\tilde{b}_i = f_i b_i \ . \tag{24}$$

Proof First note that

$$\begin{aligned}
(\tilde{x}_i - \tilde{\mu}_i) &\overset{(21,22)}{=} \left(\frac{1-f_i}{2} + f_i x_i\right) - \left(\frac{1-f_i}{2} + f_i \mu_i\right) \\
&= \frac{1-f_i}{2} + f_i x_i - \frac{1-f_i}{2} - f_i \mu_i \\
&= (x_i - \mu_i) f_i = f_i (x_i - \mu_i) \ ,
\end{aligned} \tag{25}$$

so that

$$\begin{aligned}
\tilde{E}(\tilde{\mathbf{x}}) &\overset{(18)}{=} -\sum_i (\tilde{x}_i - \tilde{\mu}_i)\, \tilde{b}_i - \sum_{i,j>i} (\tilde{x}_i - \tilde{\mu}_i)\, \tilde{w}_{ij}\, (\tilde{x}_j - \tilde{\mu}_j) \\
&\overset{(23,24,25)}{=} -\sum_i (x_i - \mu_i) \underbrace{f_i f_i}_{=1} b_i - \sum_{i,j>i} (x_i - \mu_i) \underbrace{f_i f_i}_{=1} w_{ij} \underbrace{f_j f_j}_{=1} (x_j - \mu_j) \\
&\overset{(18)}{=} E(\mathbf{x}) \ .
\end{aligned}$$

■

Definition 4 *Let there be two BMs with 'flip-invariant' energies according to Definition 2. The gradient of the LL ($\nabla\boldsymbol{\theta}$ or $\nabla\tilde{\boldsymbol{\theta}}$ respectively) are called 'flip-invariant' or 'invariant to flips of variables' if $E(\mathbf{x}) = \tilde{E}(\tilde{\mathbf{x}})$ 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 any learning rate η.*

We can now state the following theorem for the parameter updates.

Theorem 5 *The gradient of centered binary BMs with flip invariant energies according to Definition 2, is also invariant to flips of arbitrary variables if the corresponding offset parameters flip as well, that is if a flipped variable* $\tilde{x}_i = 1 - x_i$ *implies* $\tilde{\mu}_i = 1 - \mu_i$*.*

Proof

$$\begin{aligned}
\nabla\tilde{w}_{ij} &\overset{(19)}{=} \langle(\tilde{x}_i - \tilde{\mu}_i)(\tilde{x}_j - \tilde{\mu}_j)\rangle_d - \langle(\tilde{x}_i - \tilde{\mu}_i)(\tilde{x}_j - \tilde{\mu}_j)\rangle_m \\
&\overset{(25)}{=} \langle f_i(x_i - \mu_i)(x_j - \mu_j)f_j\rangle_d - \langle f_i(x_i - \mu_i)(x_j - \mu_j)f_j\rangle_m \\
&\overset{(19)}{=} f_i \nabla w_{ij} f_j \ , \\
\nabla\tilde{b}_i &\overset{(20)}{=} \langle(\tilde{x}_i - \tilde{\mu}_i)\rangle_d - \langle(\tilde{x}_i - \tilde{\mu}_i)\rangle_m \\
&\overset{(25)}{=} \langle f_i(x_i - \mu_i)\rangle_d - \langle f_i(x_i - \mu_i)\rangle_m, \\
&\overset{(20)}{=} f_i \nabla b_i \ .
\end{aligned}$$

Comparing these results with Equations (23) – (24) shows that the gradient underlies the same sign changes under variable flips as the parameters. Thus, it holds for the updated parameters $\tilde{\boldsymbol{\theta}} + \eta\nabla\tilde{\boldsymbol{\theta}}$ and $\boldsymbol{\theta} + \eta\nabla\boldsymbol{\theta}$ that

$$\begin{aligned}
\tilde{w}_{ij} + \eta\nabla\tilde{w}_{ij} &= f_i\left(w_{ij} + \eta\nabla w_{ij}\right) f_j \ , \\
\tilde{b}_i + \eta\nabla\tilde{b}_i &= f_i\left(b_i + \eta\nabla b_i\right) \ ,
\end{aligned}$$

showing that $E(\mathbf{x}) = \tilde{E}(\tilde{\mathbf{x}})$ is still guaranteed as follows from Theorem 3 and thus that the gradient of centered RBMs is flip-invariant according to Definition 4. ■

Theorem 3 and Theorem 5 hold for any value from zero to one for μ_i, 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 $p^*(x_i)$, since when $\mu_i = \sum_{x_i} p^*(x_i)\, x_i$, flipping X_i leads to $\tilde{\mu}_i = \sum_{x_i} p^*(x_i)(1 - x_i) = 1 - \sum_{x_i} p^*(x_i)\, x_i = 1 - \mu_i$.

Appendix B. Derivation of the Centered Gradient

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

We first show that the parameter transformation

$$\begin{aligned}
w_{ij} &= \hat{w}_{ij} \ , && (26) \\
b_i &= \hat{b}_i + \sum_{j \neq i} \hat{w}_{ij}\mu_j \ , && (27)
\end{aligned}$$

allows to transform a normal binary BM with energy $\hat{E}(\mathbf{x}) = -\sum_i x_i \hat{b_i} - \sum_{i,j>i} x_i \hat{w}_{ij} x_j$ into a centered binary BM with energy $E(\mathbf{x}) = -\sum_i (x_i - \mu_i)\, b_i - \sum_{i,j>i} (x_i - \mu_i)\, w_{ij}\, (x_j - \mu_j)$

and *vice versa* such that $E(\mathbf{x}) = \hat{E}(\mathbf{x}) + const$ is guaranteed for all $\mathbf{x} \in \{0,1\}^n$ and thus that the modeled distribution stays the same.

$$\begin{aligned}
E(\mathbf{x}) &= -\sum_i (x_i - \mu_i)\, b_i - \sum_{i,j>i} (x_i - \mu_i)\, w_{ij}\, (x_j - \mu_j) \\
&\overset{(26,27)}{=} -\sum_i (x_i - \mu_i) \left(\hat{b}_i + \sum_{j \neq i} \hat{w}_{ij}\mu_j \right) - \sum_{i,j>i} (x_i - \mu_i)\, \hat{w}_{ij}\, (x_j - \mu_j) \\
&= -\sum_i x_i \hat{b}_i - \sum_i x_i \sum_{j\neq i} \hat{w}_{ij}\mu_j + \sum_i \mu_i \hat{b}_i + \mu_i \sum_{j \neq i} \hat{w}_{ij}\mu_j \\
&\quad - \sum_{i,j>i} x_i \hat{w}_{ij} x_j + \sum_{i,j>i} x_i \hat{w}_{ij}\mu_j + \sum_{i,j>i} \mu_i \hat{w}_{ij} x_j - \sum_{i,j>i} \mu_i \hat{w}_{ij}\mu_j \\
&= \underbrace{-\sum_i x_i \hat{b}_i - \sum_{i,j>i} x_i \hat{w}_{ij} x_j}_{\hat{E}(\mathbf{x})} \underbrace{+ \sum_i \mu_i \hat{b}_i + \mu_i \sum_{j\neq i} \hat{w}_{ij}\mu_j - \sum_{i,j>i} \mu_i \hat{w}_{ij}\mu_j}_{const} \\
&\quad -\sum_i x_i \sum_{j\neq i} \hat{w}_{ij}\mu_j + \sum_{i,j>i} x_i \hat{w}_{ij}\mu_j + \sum_{i,j>i} \mu_i \hat{w}_{ij} x_j \\
&= \hat{E}(\mathbf{x}) + const - \sum_i x_i \sum_{j\neq i} \hat{w}_{ij}\mu_j + \sum_{i,j>i} x_i \hat{w}_{ij}\mu_j + \sum_{j,i>j} \mu_j \hat{w}_{ij} x_i \\
&= \hat{E}(\mathbf{x}) + const - \sum_i x_i \sum_{j\neq i} \hat{w}_{ij}\mu_j + \sum_{i,j\neq i} x_i \hat{w}_{ij}\mu_j \\
&= \hat{E}(\mathbf{x}) + const \ .
\end{aligned}$$

Updating the parameters of the centered BM according to Equations (19) – (20) with a learning rate η leads to an updated set of parameters w'_{ij}, b'_i, given by

$$w'_{ij} \overset{(19)}{=} w_{ij} + \eta(\langle (x_i - \mu_i)(x_j - \mu_j)\rangle_d - \langle (x_i - \mu_i)(x_j - \mu_j)\rangle_m) \ , \tag{28}$$

$$b'_i \overset{(20)}{=} b_i + \eta(\langle x_i \rangle_d - \langle x_i \rangle_m) \ . \tag{29}$$

One can now transform the updated centered BM back to a normal BM by applying the inverse transformation to the updated parameters, which finally leads to the centered gradient.

$$\begin{aligned}
\hat{w}'_{ij} &\overset{(26)}{=} w'_{ij} \\
&\overset{(26,28)}{=} \hat{w}_{ij} + \eta \underbrace{(\langle (x_i - \mu_i)(x_j - \mu_j)\rangle_d - \langle (x_i - \mu_i)(x_j - \mu_j)\rangle_m)}_{=\nabla_c \hat{w}_{ij}} \ ,
\end{aligned} \tag{30}$$

$$\begin{aligned}
\hat{b}'_i &\overset{(27)}{=} b'_i - \sum_{j\neq i} \hat{w}'_{ij}\mu_j \\
&\overset{(29,30)}{=} b_i + \eta(\langle x_i\rangle_d - \langle x_i\rangle_m) - \sum_{j\neq i}(\hat{w}_{ij} + \eta\nabla_c\hat{w}_{ij})\mu_j \\
&\overset{(27)}{=} \hat{b}_i + \sum_{j\neq i}\hat{w}_{ij}\mu_j + \eta(\langle x_i\rangle_d - \langle x_i\rangle_m) - \sum_{j\neq i}\hat{w}_{ij}\mu_j - \sum_{j\neq i}\eta\nabla_c\hat{w}_{ij}\mu_j \\
&= \hat{b}_i + \underbrace{\eta(\langle x_i\rangle_d - \langle x_i\rangle_m - \sum_{j\neq i}\nabla_c\hat{w}_{ij}\mu_j)}_{\overset{(14)}{=}\nabla_c\hat{b}_i} \quad . \qquad (31)
\end{aligned}$$

In case of an RBM the visible units are only connected to hidden units and *vice versa* such that the sum in Equation (31) either sums over all visible units or all hidden units leading to the centered gradient for RBMs given by Equations (13) – (15) (see Melchior et al., 2013, for a detailed derivation for RBMs).

Appendix C. Enhanced Gradient as a Special Case of the Centered Gradient

In the following we show that the enhanced gradient can be derived as a special case of the centered gradient. We show the equivalence for RBMs since the enhanced gradient was originally proposed for RBMs. However, the derivations for BMs and DBMs are analogous.

By setting $\boldsymbol{\mu} = \frac{1}{2}(\langle\mathbf{x}\rangle_d + \langle\mathbf{x}\rangle_m)$ and $\boldsymbol{\lambda} = \frac{1}{2}(\langle\mathbf{h}\rangle_d + \langle\mathbf{h}\rangle_m)$ we get

$$\begin{aligned}
\nabla_c\mathbf{W} &\overset{(13)}{=} \langle(\mathbf{x}-\boldsymbol{\mu})(\mathbf{h}-\boldsymbol{\lambda})^T\rangle_d - \langle(\mathbf{x}-\boldsymbol{\mu})(\mathbf{h}-\boldsymbol{\lambda})^T\rangle_m \\
&= \langle\mathbf{x}\mathbf{h}^T\rangle_d - \langle\mathbf{x}\rangle_d\boldsymbol{\lambda}^T - \boldsymbol{\mu}\langle\mathbf{h}^T\rangle_d + \boldsymbol{\mu}\boldsymbol{\lambda}^T - \langle\mathbf{x}\mathbf{h}^T\rangle_m + \langle\mathbf{x}\rangle_m\boldsymbol{\lambda}^T + \boldsymbol{\mu}\langle\mathbf{h}^T\rangle_m - \boldsymbol{\mu}\boldsymbol{\lambda}^T \\
&= \langle\mathbf{x}\mathbf{h}^T\rangle_d - \frac{1}{2}\langle\mathbf{x}\rangle_d(\langle\mathbf{h}\rangle_d + \langle\mathbf{h}\rangle_m)^T - \frac{1}{2}(\langle\mathbf{x}\rangle_d + \langle\mathbf{x}\rangle_m)\langle\mathbf{h}^T\rangle_d \\
&\quad -\langle\mathbf{x}\mathbf{h}^T\rangle_m + \frac{1}{2}\langle\mathbf{x}\rangle_m(\langle\mathbf{h}\rangle_d + \langle\mathbf{h}\rangle_m)^T + \frac{1}{2}(\langle\mathbf{x}\rangle_d + \langle\mathbf{x}\rangle_m)\langle\mathbf{h}^T\rangle_m \\
&= \langle\mathbf{x}\mathbf{h}^T\rangle_d - \frac{1}{2}\langle\mathbf{x}\rangle_d\langle\mathbf{h}^T\rangle_d - \frac{1}{2}\langle\mathbf{x}\rangle_d\langle\mathbf{h}^T\rangle_m - \frac{1}{2}\langle\mathbf{x}\rangle_d\langle\mathbf{h}^T\rangle_d - \frac{1}{2}\langle\mathbf{x}\rangle_m\langle\mathbf{h}^T\rangle_d \\
&\quad -\langle\mathbf{x}\mathbf{h}^T\rangle_m + \frac{1}{2}\langle\mathbf{x}\rangle_m\langle\mathbf{h}^T\rangle_d + \frac{1}{2}\langle\mathbf{x}\rangle_m\langle\mathbf{h}^T\rangle_m + \frac{1}{2}\langle\mathbf{x}\rangle_d\langle\mathbf{h}^T\rangle_m + \frac{1}{2}\langle\mathbf{x}\rangle_m\langle\mathbf{h}^T\rangle_m \\
&= \langle\mathbf{x}\mathbf{h}^T\rangle_d - \langle\mathbf{x}\rangle_d\langle\mathbf{h}^T\rangle_d - \langle\mathbf{x}\mathbf{h}^T\rangle_m + \langle\mathbf{x}\rangle_m\langle\mathbf{h}^T\rangle_m \\
&= \langle(\mathbf{x}-\langle\mathbf{x}\rangle_d)(\mathbf{h}-\langle\mathbf{h}\rangle_d)^T\rangle_d - \langle(\mathbf{x}-\langle\mathbf{x}\rangle_m)(\mathbf{h}-\langle\mathbf{h}\rangle_m)^T\rangle_m \\
&\overset{(1)}{=} \nabla_e\mathbf{W} \quad , \qquad (32)
\end{aligned}$$

and for the derivatives with respect to the bias parameters follows directly that

$$\begin{aligned}
\nabla_c\mathbf{b} &\overset{(14,32)}{=} \langle\mathbf{x}\rangle_d - \langle\mathbf{x}\rangle_m - \nabla_e\mathbf{W}\boldsymbol{\lambda} \\
&= \langle\mathbf{x}\rangle_d - \langle\mathbf{x}\rangle_m - \nabla_e\mathbf{W}\frac{1}{2}(\langle\mathbf{h}\rangle_d + \langle\mathbf{h}\rangle_m) \\
&\overset{(2)}{=} \nabla_e\mathbf{b} \quad ,
\end{aligned}$$

$$
\begin{aligned}
\nabla_c \mathbf{c} &\overset{(15,32)}{=} \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m - \nabla_e \mathbf{W}^T \boldsymbol{\mu} \\
&= \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m - \nabla_e \mathbf{W}^T \frac{1}{2} \left(\langle \mathbf{x} \rangle_d + \langle \mathbf{x} \rangle_m \right) \\
&\overset{(3)}{=} \nabla_e \mathbf{c} \ .
\end{aligned}
$$

Appendix D. Comparison of the Different MNIST Variants

In this section we compare the performance of normal and centered RBMs on different variants of the *MNIST* data set (LeCun et al., 1998). The data set is provided at `http://yann.lecun.com/exdb/mnist/` and consists of pixel values in the range of 0 to 255.

Algorithm-η	dd_s^b			00		
MNIST-Probabilities						
CD-1-0.1	**-156.46**	±2.48	(-157.1)	-168.54	±3.99	(-168.5)
CD-1-0.05	**-155.60**	±2.31	(-157.0)	-172.34	±2.53	(-172.4)
CD-1-0.01	**-155.40**	±2.24	(-155.4)	-171.66	±1.85	(-173.3)
PCD-1-0.1	**-147.42**	±1.06	(-148.3)	-166.70	±3.00	(-183.1)
PCD-1-0.05	**-145.42**	±1.24	(-145.7)	-160.54	±4.76	(-162.5)
PCD-1-0.01	**-144.78**	±0.43	(-144.8)	-148.85	±0.99	(-149.0)
PT_{20}-0.01	**-144.92**	±0.57	(-144.9)	-150.97	±2.45	(-151.1)
MNIST-Sampled						
CD-1-0.1	**-155.21**	±1.63	(-157.8)	-169.95	±2.61	(-170.1)
CD-1-0.05	**-154.84**	±2.01	(-156.2)	-170.84	±2.92	(-171.0)
CD-1-0.01	**-154.95**	±1.96	(-155.0)	-171.90	±2.23	(-173.6)
PCD-1-0.1	**-145.03**	±1.13	(-146.0)	-151.04	±3.73	(-152.3)
PCD-1-0.05	**-144.10**	±1.16	(-144.1)	-147.53	±1.61	(-147.9)
PCD-1-0.01	**-143.86**	±0.63	(-143.9)	-146.89	±0.66	(-147.0)
PT_{20}-0.01	**-144.01**	±0.63	(-144.2)	-149.18	±1.51	(-149.2)
MNIST-Threshold						
CD-1-0.1	**-151.94**	±2.38	(-154.0)	-168.96	±4.42	(-169.2)
CD-1-0.05	**-151.84**	±3.72	(-152.5)	-170.39	±3.01	(-170.5)
CD-1-0.01	**-151.53**	±2.17	(-151.5)	-169.15	±2.01	(-171.7)
PCD-1-0.1	**-143.56**	±2.43	(-144.8)	-164.34	±3.30	(-187.2)
PCD-1-0.05	**-141.00**	±1.54	(-142.0)	-156.87	±4.15	(-160.3)
PCD-1-0.01	**-139.61**	±0.91	(-139.8)	-143.70	±0.95	(-143.8)
PT_{20}-0.01	**-140.20**	±1.08	(-140.3)	-146.75	±1.96	(-147.0)

Table 13: Maximum average LL on the test data for centered and normal RBMs with 16 hidden units trained on the three different variants of MNIST averaged over 25 trials. The models were trained for 100 epochs with a batch size of 100 and in the case of dd_s^b a sliding factor of 0.01 was used.

ALGORITHM-η	dd^b_s			00		
MNIST-Probabilities						
PCD-1-0.01	<u>-100.32</u>	±1.10	(-100.3)	-111.47	±1.76	(-111.5)
PCD-1-0.005	<u>-100.04</u>	±0.98	(-100.0)	-108.76	±1.13	(-109.8)
PT_{20}-0.01	<u>-97.80</u>	±0.40	(-97.8)	-105.93	±3.31	(-106.5)
MNIST-Sampled						
PCD-1-0.01	<u>-92.00</u>	±1.79	(-92.0)	-94.95	±1.19	(-97.8)
PCD-1-0.005	<u>-90.85</u>	±0.61	(-91.4)	-94.29	±1.29	(-94.3)
PT_{20}-0.01	<u>-90.13</u>	±0.74	(-90.1)	-98.35	±2.68	(-99.8)
MNIST-Threshold						
PCD-1-0.01	<u>-78.69</u>	±1.22	(-80.1)	-101.06	±3.37	(-107.1)
PCD-1-0.005	<u>-75.93</u>	±0.69	(-75.9)	-94.72	±2.07	(-97.8)
PT_{20}-0.01	<u>-74.77</u>	±1.08	(-75.2)	-87.79	±5.02	(-87.8)

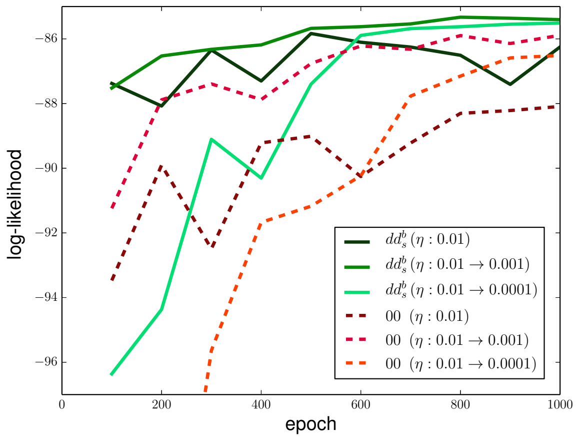
Table 14: Maximum average LL on the test data for centered and normal RBMs with 500 hidden units trained on the three different variants of MNIST averaged over 10 trials. The models were trained for 200 epochs with a batch size of 100 and in the case of dd^b_s a sliding factor of 0.01 was used.

Since the data set is not binary it has to be preprocessed when used with binary RBMs. In a first step the values are usually normalized to lie in $[0, 1]$. In different studies the normalized values are then either used directly as input, or the data set is binarized using a threshold of 0.5 or by sampling according to the normalized values (which are treated as the probabilities for the variables being in state 1). Here, we refer to these three different binarized variants of *MNIST* as *MNIST-Probabilities*, *MNIST-Threshold*, and *MNIST-Sampled*, respectively.

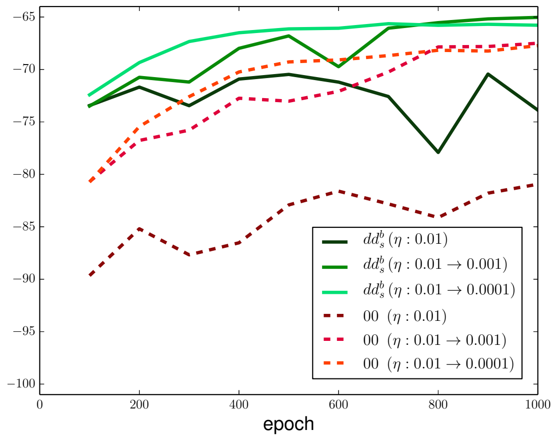
Since it is often not mentioned how the data set is binarized it is hard to compare the LL values reported for *MNIST* experiments across studies. Furthermore, since the results are mainly given for big models the LL is usually approximated using AIS, which easily overestimates the LL if the setup is not chosen properly. In our experience using a reasonable base partition function such as one that is based on the MLE for zero weights (Salakhutdinov and Murray, 2008) is crucial for AIS not to dramatically overestimate the LL for RBMs/DBMs. In this work, we ensured that when given the same pseudo random number generator, our AIS implementation and the *MNIST-Sampled* data set were the same as used by Salakhutdinov and Murray (2008).

As a baseline we trained normal and centered RBMs with 16 hidden units on the three different *MNIST* variants for 100 epochs using the setup described in Section 6. The maximum average of the exact LL values for the test data for dd^b_s and 00 are given in Table 13. Although the maximum reached values are quite different for the different *MNIST* variants, centered RBMs outperformed normal RBMs on all three data sets.

We continued by training normal and centered RBMs with 500 hidden units on the three different data sets for 200 epochs. The setup for training and evaluating the models was



(a) *MNIST-Sampled*



(b) *MNIST-Threshold*

Figure 18: Evolution of the LL of single trials on the test data of (a) *MNIST-Sampled* and (b) *MNIST-Threshold* for dd_s^b and 00 with 500 hidden units. The models were trained for 1000 epochs with a weight decay of 0.0002 and a momentum of 0.9 that was reduced to 0.5 after 5 epochs. The learning rate was either fixed to 0.01, decayed from 0.01 to 0.001 or from 0.01 to 0.0001 over 1000 epochs.

the same as described in Section 6. The maximum average LL values are given in Table 14, showing again that the reached LL on the different *MNIST* variants is quite different and that centered RBMs reach better values than normal RBMs on all three data sets.

In our experience the use of weight decay, momentum and an annealing learning rate is crucial to reach a LL that is comparable to the value of -86.34 reported by Salakhutdinov and Murray (2008). We therefore trained centered RBMs (dd_s^b) and normal RBMs (00) using PCD-25 for the extensive amount of 1000 epochs (600000 gradient updates). In addition a weight decay of 0.0002, and a momentum of 0.9 that was set to 0.5 after 5 epochs was used. The learning rate was fixed to 0.01 or either decayed from 0.01 to 0.001 or from 0.01 to 0.0001 over 1000 epochs.

The average LL on the test data of *MNIST-Sampled* for single trials are shown in Figure 18(a). It can be seen that a decaying learning rate is crucial for 00, which only reaches a value of -88.0 without decaying learning rate but -85.9 with a learning rate that decayed from 0.01 to 0.001. dd_s^b reaches a LL value of -86.2 without and -85.5 with a learning rate that decayed from 0.01 to 0.001. Although the difference between normal and centered RBMs become smaller with weight decay, dd_s^b still reaches a higher LL, seems to be less depended on the learning rate schedule and allows faster learning. We performed the same experiments also for the *MNIST-Threshold* data set. The results are shown in Figure 18(b), which show qualitatively the same results. The difference between 00 and dd_s^b is more prominent on the *MNIST-Threshold* where 00 reaches a LL value of -67.5 and dd_s^b reaches -65.0, which can also be seen from Table 14.

References

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, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19:153, 2007.

C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

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*, 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 *Proceedings of the International Conference on Machine Learning*, pages 105–112. Omnipress, 2011.

K. Cho, T. Raiko, and A. Ilin. Gaussian-Bernoulli deep Boltzmann machine. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–7. IEEE, 2013a.

K. Cho, T. Raiko, and A. Ilin. Enhanced gradient for training restricted Boltzmann machines. *Neural Computation*, 25(3):805–831, 2013b.

G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Dellaleau. Tempered Markov Chain Monte Carlo for training of restricted Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 9, pages 145–152, 2010.

G. Desjardins, R. Pascanu, A. Courville, and Y. Bengio. Metric-free natural gradient for joint-training of Boltzmann machines. *Computing Research Repository*, 2013.

A. Fischer. *Training Restricted Boltzmann Machines.* PhD thesis, University of Copenhagen, 2014.

A. Fischer and C. Igel. Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines. In *Proceedings of the International Conference on Artificial Neural Networks*, volume 6354 of *LNCS*, pages 208–217. Springer-Verlag, 2010.

A. Fischer and C. Igel. Bounding the bias of contrastive divergence learning. *Neural Computation*, 23(3):664–673, 2011.

A. Fischer and C. Igel. Training restricted Boltzmann machines: An introduction. *Pattern Recognition*, 47:25–39, 2014.

A. Fischer and C. Igel. A bound for the convergence rate of parallel tempering for sampling restricted Boltzmann machines. *Theoretical Computer Science*, 2015.

R. Grosse and R. Salakhudinov. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In *Proceedings of the International Conference on Machine Learning*, 2015.

G. E. Hinton. A practical guide to training restricted Boltzmann machines. Technical report, Department of Computer Science, University of Toronto, 2010.

G. E. Hinton and R. Salakhutdinov. A better way to pretrain deep Boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2447–2455. Curran Associates, Inc., 2012.

G. E. Hinton, O. Simon, and T. Yee-Whye. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Computing Research Repository*, abs/1502.03167, 2015. URL `http://arxiv.org/abs/1502.03167`.

H. J. Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 15, pages 29–37, 2011.

H. Larochelle, Y. Bengio, and J. Turian. Tractable multivariate binary density estimation and the restricted Boltzmann forest. *Neural Computation*, 22:22852307, 2010.

Y. LeCun, L. Bottou, G. Orr, and K. R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science, page 546. Springer Berlin / Heidelberg, 1998.

M. Lingenheil, R. Denschlag, G. Mathias, and P. Tavan. Efficiency of exchange schemes in replica exchange. *Chemical Physics Letters*, 478:80–84, 2009.

D. MacKay. *Information Theory, Inference, and Learning Algorithm.* Cambridge University Press, Cambridge, 2003.

B. Marlin, K. Swersky, B. Chen, and N. de Freitas. Inductive principles for learning restricted Boltzmann machines. In *Artificial Intelligence and Statistics Conference*, 2010.

J. Melchior, A. Fischer, and L. Wiskott. How to center binary restricted Boltzmann machines. *Computing Research Repository*, 2013. URL `http://arxiv.org/abs/1311.1354`.

G. Montavon and K. R. Müller. Deep Boltzmann machines and the centering trick. *Lecture Notes in Computer Science*, 7700:621–637, 2012.

A. Ng. Sparse autoencoder. Technical report, Stanford University, 2011.

Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Computing Research Repository*, 2011.

B. A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

C. Poultney, S. Chopra, Y. L. Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137–1144, 2006.

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.

S. Rifai, P. Vincent, X Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the International Conference on Machine Learning*, pages 833–840, 2011.

D. Rumelhart, G. E. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986a.

D. Rumelhart, J. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, 1986b.

R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *Artificial Intelligence and Statistics Conference*, volume 5, pages 448–455, Clearwater Beach, Florida, USA, 2009.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the International Conference on Machine Learning*, pages 872–879, New York, 2008. ACM.

N. Schraudolph. Centering neural network gradient factors. In *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 207–226. Springer Verlag, Berlin, 1998.

H. Schulz, A. Müller, and S. Behnke. Investigating convergence of restricted Boltzmann machine learning. In *Proceedings of the NIPS 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.

P. Smolensky. *Information Processing in Dynamical Systems: Foundations of Harmony Theory.* MIT Press, Cambridge, MA, USA, 1986.

R. H. Swendsen and J. S. Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57(21):2607, 1986.

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*, 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.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the International Conference on Machine Learning*, pages 1064–1071. ACM, 2008.

T. Tieleman and G. E. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the International Conference on Machine Learning*, pages 1033–1040, New York, 2009. ACM.

P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

L. Younes. Maximum likelihood estimation of Gibbs fields. In *Proceedings of the Joint Conference on Spacial Statistics and Imaging*, Lecture Notes Monograph Series. Institute of Mathematical Statistics, Hayward, California, 1991.