How to Center Binary Restricted Boltzmann Machines

Jan Melchior[1], Asja Fischer[*1], Nan Wang[†1], and Laurenz Wiskott[1]

[1]Institut für Neuroinformatik, Ruhr Universität Bochum, 44780 Bochum, Germany
<forename>.<surname>@ruhr-uni-bochum.de

Abstract

It has recently been shown that subtracting the mean from the visible as well as the hidden variables of deep Boltzmann machines leads to better conditioned optimization problems and improves some aspects of model performance. In this work we analyze binary restricted Boltzmann machines, where centering is done by subtracting offset values from visible and hidden variables. We show analytically that (i) the expected performance of centered binary restricted Boltzmann machines is invariant under simultaneous flip of data and offsets, for any offset value in the range of zero to one, and (ii) using the 'enhanced gradient' is equivalent to setting the offset values to the average over model and data mean. Our results also generalize to deep Boltzmann machines. Numerical simulations suggest that (i) optimal generative performance is archived by subtracting mean values from visible as well as hidden variables, (ii) the enhanced gradient suffers from divergence more often than other centering variants, (iii) learning is stabilized if a sliding average over the batch means is used for the offset values instead of the current batch mean, this also prevents the enhanced gradient from divergence.

1 Introduction

In the last decade Restricted Boltzmann Machines (RBMs) got in 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 given the training data. Since the gradient is intractable, it is approximated by Gibbs samples from a Markov chain iterated only for a few steps.

[*]Asja Fischer was supported by the German Federal Ministry of Education and Research within the National Network Computational Neuroscience under grant number 01GQ0951

[†]Nan Wang was supported by the fellowship from International Graduate School of Neuroscience (IGSN), Ruhr-Universität Bochum.

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). To overcome the divergence problem, Desjardins et al. (2010) proposed to use parallel tempering, which is an advanced sampling method that leads to a faster mixing Markov chain and thus to better approximations of the LL gradient. Secondly, the learning process is not invariant to the data representation. For example training an RBM on the *MNIST* dataset leads to a better model than training it on *1-MNIST* (the dataset generated by flipping each bit in *MNIST*). This is due to missing invariance properties of the gradient with respect to these flip transformations and not due to the model's capacity, since an RBM trained on *MNIST* can be transformed in such a way that it models *1-MNIST* with the same LL.

Recently, two approaches have been introduced that address the invariance problem. The enhanced gradient (Cho et al., 2011, 2013a) has been designed as an invariant alternative to the true LL gradient of binary RBMs and derived by calculating a weighted average over the gradients one gets by applying any possible bit flip combination on the dataset. Cho et al. (2011) have shown empirically 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, the enhanced gradient in combination with an adaptive learning rate leads to more stable training in the sense that good LL values are reached independently from the initialization of the learning rate. Tang and Sutskever (2011) 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* dataset and comparable results to those of the enhanced gradient. Removing the mean from the variables is generally known as the 'centering trick' which was originally proposed for feed forward neural networks (LeCun et al., 1998). It has recently also been applied to the hidden and visible 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 better discriminative properties and centering improves the generative properties of locally connected DBMs.

In this work we give a unified view on centering in RBMs, show that the enhanced gradient is a particular form of centering and analyze the different ways of choosing and approximating the offset parameters empirically. We begin with a brief overview over binary RBMs, the standard learning algorithms, and the basic ideas used to construct the enhanced gradient in section 2. In section 3 we discus the theoretical properties of centered RBMs and show that the enhanced gradient is a particular form of centering.

Finally, we empirically analyze the training of centered RBMs with different offset parameters, sampling methods, and learning rates in section 5.

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 $\mathbf{x} = (x_0, ..., x_N)$ and $\mathbf{h} = (h_0, ..., h_M)$, respectively. Since an RBM is a Markov random field, its joint probability distribution

is given by a Gibbs distribution,

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

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

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{x}^T \mathbf{b} - \mathbf{c}^T \mathbf{h} - \mathbf{x}^T \mathbf{W} \mathbf{h}, \tag{2}$$

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 $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{H}}$ respectively, is given by

$$Z = \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} \mathrm{e}^{-E(\tilde{\mathbf{x}}, \tilde{\mathbf{h}})}. \tag{3}$$

RBM training is usually based on gradient ascent using approximations of the log-likelihood gradient

$$\nabla \boldsymbol{\theta} = \frac{\partial \left\langle \log\left(p(\mathbf{x}|\boldsymbol{\theta})\right)\right\rangle_d}{\partial \boldsymbol{\theta}} = -\left\langle \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_d + \left\langle \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_m, \tag{4}$$

where $\langle \cdot \rangle_m$ is the expectation under $p(\mathbf{h}, \mathbf{x})$ and $\langle \cdot \rangle_d$ is the one 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 log-likelihood with respect to $\boldsymbol{\theta}$ in order to be consistent with the notation in (Cho et al., 2011). For binary RBMs the gradient becomes $\nabla \mathbf{W} = \langle \mathbf{x}\mathbf{h}^T \rangle_d - \langle \mathbf{x}\mathbf{h}^T \rangle_m$, $\nabla \mathbf{b} = \langle \mathbf{x} \rangle_d - \langle \mathbf{x} \rangle_m$, $\nabla \mathbf{c} = \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m$.

Common RBM training methods approximate $\langle \cdot \rangle_m$ by samples gained by different Markov chain Monte Carlo methods. Sampling (k steps) from a Gibbs chain initialized with a data sample yields the Contrastive Divergence (CD-k) (Hinton et al., 2006) algorithm. In Persistent Contrastive Divergence (PCD-k) (Tieleman, 2008) the chain is not reinitialized after parameter updates, which has been reported to lead to better approximations if the learning rate is chosen sufficiently small. The advanced sampling method Parallel Tempering (PT$_c$) introduces c additional "tempered" Gibbs chains corresponding to smoothed versions of $p(\mathbf{x}, \mathbf{h})$ and allows samples to swap between chains. PT$_c$ increases the mixing rate and has been reported to achieve better approximations than CD-k and PCD-k (Desjardins et al., 2010), but it also has a higher computational cost.

2.1 Enhanced Gradient

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

When transforming the state $(\mathbf{x}, \mathbf{h})$ of a binary RBM by flipping some of its variables (i.e $\tilde{x_i} = 1 - x_i$ 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 $E(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta}) = E(\tilde{\mathbf{x}}, \tilde{\mathbf{h}}|\tilde{\boldsymbol{\theta}}) + const$ and thus $p(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta}) = p(\tilde{\mathbf{x}}, \tilde{\mathbf{h}}|\tilde{\boldsymbol{\theta}})$ holds. However, if we update the parameters of

the transformed model based on the corresponding log-likelihood gradient to $\tilde{\boldsymbol{\theta}}' = \tilde{\boldsymbol{\theta}} + \eta\nabla\tilde{\boldsymbol{\theta}}$ and apply the inverse parameter transformation to $\tilde{\boldsymbol{\theta}}'$, the result will differ from $\boldsymbol{\theta}' = \boldsymbol{\theta} + \eta\nabla\boldsymbol{\theta}$. The described procedure of transforming, updating, and transforming back can be regarded as a different way to update $\boldsymbol{\theta}$.

Following this line of thought there exist 2^{N+M} different parameter updates corresponding to the 2^{N+M} possible binary flips of $(\mathbf{x}, \mathbf{h})$. 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 weight is given by

$$\nabla_e \mathbf{W} = \langle(\mathbf{x} - \langle\mathbf{x}\rangle_d)(\mathbf{h} - \langle\mathbf{h}\rangle_d)^T\rangle_d - \langle(\mathbf{x} - \langle\mathbf{x}\rangle_m)(\mathbf{h} - \langle\mathbf{h}\rangle_m)^T\rangle_m \quad (5)$$

$$\nabla_e \mathbf{b} = \langle\mathbf{x}\rangle_d - \langle\mathbf{x}\rangle_m - \nabla_e \mathbf{W}\frac{1}{2}\left(\langle\mathbf{h}\rangle_d + \langle\mathbf{h}\rangle_m\right) \quad (6)$$

$$\nabla_e \mathbf{c} = \langle\mathbf{h}\rangle_d - \langle\mathbf{h}\rangle_m - \nabla_e \mathbf{W}^T\frac{1}{2}\left(\langle\mathbf{x}\rangle_d + \langle\mathbf{x}\rangle_m\right) \quad (7)$$

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

3 Centered Restricted Boltzmann Machines

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

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

$$E(\mathbf{x}, \mathbf{h}) = -(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{b} - \mathbf{c}^T(\mathbf{h} - \boldsymbol{\lambda}) - (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{W}(\mathbf{h} - \boldsymbol{\lambda}) \,. \quad (8)$$

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

$$p(x_i = 1|\mathbf{h}) = \text{sigm}(\mathbf{w_{i*}}(\mathbf{h} - \boldsymbol{\lambda}) + b_i) \,, \quad (9)$$

$$p(h_j = 1|\mathbf{x}) = \text{sigm}((\mathbf{x} - \boldsymbol{\mu})^T \mathbf{w_{*j}} + c_j) \,, \quad (10)$$

where $\text{sigm}(\cdot)$ 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 log-likelihood gradient now takes the form

$$\begin{aligned}
\nabla \mathbf{W} &= \langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_d - \langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_m \,, & (11)\\
\nabla \mathbf{b} &= \langle \mathbf{x} - \boldsymbol{\mu} \rangle_d - \langle \mathbf{x} - \boldsymbol{\mu} \rangle_m = \langle \mathbf{x} \rangle_d - \langle \mathbf{x} \rangle_m \,, & (12)\\
\nabla \mathbf{c} &= \langle \mathbf{h} - \boldsymbol{\lambda} \rangle_d - \langle \mathbf{h} - \boldsymbol{\lambda} \rangle_m = \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m \,. & (13)
\end{aligned}$$

$\nabla \mathbf{b}$ 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 that the gradient of the centered RBM is invariant to flip transformations if a flip x_i to $1 - x_i$ implies a change of μ_i to $1 - \mu_i$ and a flip h_j to $1 - h_j$ implies a change of λ_j to $1 - \lambda_j$. This holds in particular for $\mu_i = 0.5$, $\lambda_j = 0.5$ and any expectation over x_i and h_j under any distribution. Note, that the invariance proof also generalizes to DBMs

If we set $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ to the expectation 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 of a centered RBM needs to update the offset values and transform the RBM parameters such that the modeled probability distribution stays the same.

An RBM with offsets $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ can be transformed to an RBM with offsets $\boldsymbol{\mu}'$ and $\boldsymbol{\lambda}'$ by

$$\begin{aligned}
\mathbf{W}' &= \mathbf{W} \,, & (14)\\
\mathbf{b}' &= \mathbf{b} + \mathbf{W} \left(\boldsymbol{\lambda}' - \boldsymbol{\lambda} \right) \,, & (15)\\
\mathbf{c}' &= \mathbf{c} + \mathbf{W}^T \left(\boldsymbol{\mu}' - \boldsymbol{\mu} \right) \,, & (16)
\end{aligned}$$

such that $E(\mathbf{x}, \mathbf{h} | \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = E(\mathbf{x}, \mathbf{h} | \boldsymbol{\theta}', \boldsymbol{\mu}', \boldsymbol{\lambda}') + const$, is guaranteed.

3.1 Centered Gradient

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

$$\begin{aligned}
\nabla_c \mathbf{W} &= \langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_d - \langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{h} - \boldsymbol{\lambda})^T \rangle_m \,, & (17)\\
\nabla_c \mathbf{b} &= \langle \mathbf{x} \rangle_d - \langle \mathbf{x} \rangle_m - \nabla_c \mathbf{W} \boldsymbol{\lambda} \,, & (18)\\
\nabla_c \mathbf{c} &= \langle \mathbf{h} \rangle_d - \langle \mathbf{h} \rangle_m - \nabla_c \mathbf{W}^T \boldsymbol{\mu} \,. & (19)
\end{aligned}$$

Notice that by setting $\boldsymbol{\mu} = \frac{1}{2} \left(\langle \mathbf{x} \rangle_d + \langle \mathbf{x} \rangle_m \right)$ and $\boldsymbol{\lambda} = \frac{1}{2} \left(\langle \mathbf{h} \rangle_d + \langle \mathbf{h} \rangle_m \right)$ the centered gradient becomes equal to the enhanced gradient. Thus, it becomes clear that the enhanced gradient is a special case of centering. This can also be concluded from the derivation of the enhanced gradient for Gaussian visible variables in (Cho et al., 2013b). The enhanced gradient has been designed such that the weight updates become the difference of the covariances between one hidden and one visible variable

Algorithm 1 Training RBMs using the centered gradient

Initialize $\mathbf{W}$ ($i.e.$ $\mathbf{W} \leftarrow \mathcal{N}(0, 0.01)^{N \times M}$)
Initialize $\boldsymbol{\mu}, \boldsymbol{\lambda}$ ($i.e.$ $\boldsymbol{\mu} \leftarrow \langle \mathbf{X}_d \rangle, \boldsymbol{\lambda} \leftarrow \mathbf{0.5}$)
Initialize $\mathbf{b}, \mathbf{c}$ ($i.e.$ $\mathbf{b} \leftarrow \text{sigm}^{-1}(\boldsymbol{\mu}), \mathbf{c} \leftarrow \mathbf{0}$)
repeat
  for all batches $\mathbf{X}_d$ **do**
    Calculate $\mathbf{H}_d = p(\mathbf{H} = 1 | \mathbf{X}_d)$
    Sample $\mathbf{X}_m$ *from binary RBM*
    Calculate $\mathbf{H}_m = p(\mathbf{H} = 1 | \mathbf{X}_m)$
    Estimate $\boldsymbol{\mu}_{new}$ ($i.e.$ $\boldsymbol{\mu}_{new} \leftarrow \langle \mathbf{X}_d \rangle$)
    Estimate $\boldsymbol{\lambda}_{new}$ ($i.e.$ $\boldsymbol{\lambda}_{new} \leftarrow \langle \mathbf{H}_d \rangle$)
    $\boldsymbol{\mu} \leftarrow (1 - \nu_\mu)\boldsymbol{\mu} + \nu_\mu \boldsymbol{\mu}_{new}$
    $\boldsymbol{\lambda} \leftarrow (1 - \nu_\lambda)\boldsymbol{\lambda} + \nu_\lambda \boldsymbol{\lambda}_{new}$
    $\nabla_c \mathbf{W} \leftarrow \langle (\mathbf{X}_d - \boldsymbol{\mu})(\mathbf{H}_d - \boldsymbol{\lambda})^T \rangle$
      $- \langle (\mathbf{X}_m - \boldsymbol{\mu})(\mathbf{H}_m - \boldsymbol{\lambda})^T \rangle$
    $\nabla_c \mathbf{b} \leftarrow \langle \mathbf{X}_d \rangle - \langle \mathbf{X}_m \rangle - \nabla_c \mathbf{W} \boldsymbol{\lambda}$
    $\nabla_c \mathbf{c} \leftarrow \langle \mathbf{H}_d \rangle - \langle \mathbf{H}_m \rangle - \nabla_c \mathbf{W}^T \boldsymbol{\mu}$
    $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla_c \mathbf{W}$
    $\mathbf{b} \leftarrow \mathbf{b} + \eta \nabla_c \mathbf{b}$
    $\mathbf{c} \leftarrow \mathbf{c} + \eta \nabla_c \mathbf{c}$
  end for
until stopping criteria is met

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, the choice of these offsets results in different update rules for the bias parameters.

Training a normal binary RBM based on the centered gradient is equivalent to training a centered RBM and finally transforming it to a normal binary RBM. Algorithm 1 shows pseudo code for training a normal binary RBM using the centered gradient, which can easily be rewritten for training a centered RBM. Note that the update of the offsets – as it is the case when using the centered gradient – is performed before the gradient is calculated. This is in contrast to the algorithm for centered DBMs proposed in (Montavon and Müller, 2012), where the update of the offsets and the reparameterization follows after the gradient update. This implicates that the estimates of the offsets in one learning iteration are based on samples gained from the model of the previous iteration. However, the proposed DBM algorithm smoothes the offset estimations by a sliding average over the means of samples from many iterations, so that the choice of the sample set used for the offset estimation should be less relevant. In Algorithm 1 a sliding average is obtained if $0 < \nu < 1$ and prevented if $\nu = 1$.

3.2 Initialization of the bias parameters

Montavon and Müller (2012) have suggested that initializing the bias parameters to the

inverse sigmoid of the initial offset parameters leads to a good starting point, because it guarantees that the Boltzmann machine is initially centered. Thus, they set the offset and the bias of the visible variables initially to $\langle \mathbf{x} \rangle_d$ and $\text{sigm}^{-1}(\langle \mathbf{x} \rangle_d)$, respectively. And the hidden offset was set to $\text{sigm}(c_{init})$ for an initial hidden bias parameter $c_{init} \in \{-2, 0, 2\}$.

Suppose the weight matrix is initialized to small random values (i.e. we can assume that they are approximately zero) then the conditional probabilities (9) and (10), are approximately given by $p(x_i = 1|\mathbf{h}) \approx \text{sigm}(b_i)$ and $p(h_j = 1|\mathbf{x}) \approx \text{sigm}(c_j)$, respectively. If the visible bias is initialized to the inverse sigmoid of the data mean, the expectation of the conditional distribution under the model takes approximately the same value. We argue that it is reasonable to assume an initial mean of 0.5 for the hidden variables, if the initial weight parameters are negligible small. Therefore, 0.5 and $\text{sigm}(0.5) = 0$ should be the initialization of choice for the hidden offset and bias parameters, respectively. We claim that this initialization may also be beneficial for normal binary RBMs and centered RBMs with $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_d$ and $\boldsymbol{\lambda} = 0$. Following the same arguments as above $\langle \mathbf{x} \rangle_m$ and $\langle \mathbf{h} \rangle_m$ can also be approximated by 0.5. Thus, to guarantee that the centered RBM corresponding to the enhanced gradient is initially centered, the visible and hidden biases should be initialized to $\text{sigm}^{-1}(0.5(\langle \mathbf{x} \rangle_d + 0.5))$ and 0, respectively.

4 Experiments

As shown in the previous section the algorithms described by Montavon and Müller (2012),Tang and Sutskever (2011) and Cho et al. (2011) can all be viewed as different versions of centered RBMs. 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 a sliding average or not. In the following we analyze the effect of these differences on the learning outcome.

For simplicity we introduce the following shorthand notation. We use d to denote that the data mean $\langle \cdot \rangle_d$ is used, m for the model mean $\langle \cdot \rangle_m$, a for the average of the means $\frac{1}{2}\langle \cdot \rangle_d + \frac{1}{2}\langle \cdot \rangle_m$ and *0* if the offsets is set to zero. We indicate the choice of $\boldsymbol{\mu}$ in the first and the choice of $\boldsymbol{\lambda}$ in the second place, e.g. dd translates to $\boldsymbol{\mu} = \langle \mathbf{v} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_d$. We add a superscribed b or a to denote whether the reparameterization is performed before or after the gradient update. If a sliding factor is used a subscript s is added. Thus, we indicate the variant of Montavon and Müller (2012) by dd_s^a, the one of Cho et al. (2011) by aa^b, the data normalization in Tang and Sutskever (2011) by $d0$, and the normal binary RBM simply by *00*.

In a first set of experiments we analyze these four algorithms in terms of the evolution of the LL during training. We focus our analysis on RBMs, where one layer is small enough to guarantee that the exact LL is still tractable, to avoid approximation problems when using annealed importance sampling (Schulz et al., 2010). In a second set of experiments we analyze the effect of the initialization described in section 3.2.

We proceed with a comparison of dd_s^b and dd_s^a estimating offset values and reparameterizing the parameters before and after the gradient update, respectively. Finally we analyze the effects of using a sliding average to approximate the offset values in the

different algorithms and of using dm as a centering version with a new choice of offset parameters.

4.1 Benchmark Problems

For our analysis we consider three different benchmark problems.
The **Bars & Stripes** (MacKay, 2002) problem consists of quadratic patterns of size $N = D \times D$ that can be generated as follows. First, a vertical or horizontal orientation are chosen randomly with equal probability. Then the states of all pixels of every row or column is chosen uniformly at random. This leads to $2^{D+1} - 2$ different patterns, where the completely uniform patterns occur twice as often as the others. The dataset 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 $-N \ln(N) + 4 \ln(2)$. For our experiments we used $D = 3$ leading to an upper bound of -41.59.
The **Shifting Bar** dataset is an artificial benchmark problem we designed to be asymmetric in terms of the amount of zeros and ones in the data. For an input dimensionality N, a bar length $0 < B < N$ has to be chosen, where $\frac{B}{N}$ expresses the percentage of ones in the dataset. A position $0 \leq p < N$ is chosen uniformly at random and the states of the following B pixels are set to one, where a wrap around is used if $p+B \geq N$. The states of the remaining pixels are set to zero. This leads to N different patterns, with equal probability and an upper bound of the LL of $-N \ln(N)$. 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) database of handwritten digits has become a standard benchmark problem for RBMs. It consists of $50,000$ training, $10,000$ validation and $10,000$ testing examples of gray value handwritten digits of size 28×28. After binarization the dataset contains 13.3% ones, similar to the *Shifting Bar* problem, which in our case contains 11.1% ones. For training and evaluating the LL we used the binarized $50,000$ training examples.

4.2 Experimental Setup

The RBMs weight matrices were initialized with random values samples from a Gaussian with zero mean and a standard deviation of 0.01. If not stated otherwise the visible and hidden bias parameters were initially set to zero. In all experiments we used CD-1, PCD-1 and PT_{10} as the three common training algorithms for RBMs. Full-batch training was used for *Bars & Stripes* and *Shifting Bar* and mini-batch training with a batch size of 100 was used for *MNIST*. In each trial $50,000$ parameter updates were performed. We used 16 hidden variables when modeling *MNIST* and 4 hidden variables to model *Shifting Bar* and *Bars & Stripes*. To save computation time, the LL was calculated every 50th gradient update for *Shifting Bar* and *Bars & Stripes* dataset and every epoch (500 gradient updates) for *MNIST*.

5 Results and Discussion

All tables given in this section show the average maximum LL reached during training with different learning algorithms over 25 trials and the corresponding standard deviation. In some cases the final average LL reached at the end of training is given in parenthesis to indicate a potential divergence of the LL. In the case of *MNIST* the average LL was divided by 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.

5.1 Comparison of the standard methods

The comparison of the learning performance of the previously described algorithms dd_s^a, aa^b, $d0$, and 00 show that training a centered RBM leads to significantly higher LL values than training a normal binary RBM (see Table 1 for the results for *Bars & Stripes* and *MNIST*). Figure 1 illustrates on the *Bars & Stripes* dataset that centering both the visible and the hidden variables (dd_s^a and aa^b) compared to centering only the visible variables ($d0$) accelerates the learning and leads to a higher LL when using PT and PCD, (see Table 1). It can also be seen that all methods show divergence in combination with CD, which can be prevented for dd_s^a, $d0$, and 00 when using PT. aa however suffers from severe divergence of the LL when PT is used, which is even worse than with CD. This problem does not depend on the choice of the learning rate as indicated by the LL values reached at the end of training (given in parentheses) in Table 1. All observations have also been made for the *Shifting Bar* and the *Flipped Shifting Bar* dataset where the results can not be given because of space restriction. These results also demonstrate the flip invariance of the centering empirically. While 00 fails to model the flipped version of the dataset correctly dd_s^a, aa^b, $d0$ have approximately the same performance on the flipped and unflipped datset.

5.2 Initialization

We trained normal binary RBMs (i.e. 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 2 shows the results for the normal binary RBM trained on the *Flipped Shifting Bar* dataset, where the RBM with zero initialization failed to learn the distribution accurately. The RBMs using the inverse sigmoid initialization achieved good performance and therefore seem to be less sensitive to the "difficult" representation of the data. We also trained models using the centering versions dd, aa, and $d0$ comparing the initialization suggested in section 3.2 against the zero initialization, where we observed that the different ways to initialize had little effect on the performance. In most cases the results either show no significant difference in terms of the maximum LL between the initializations or lead to slightly better results when using the inverse sigmoid. This is in particular the case when the learning rate is small. As an example Table 3 shows the results for dd_s^a on the *Bars & Stripes* dataset. In addition, the

initialization leads to slightly faster learning, thus we used it in following experiments.

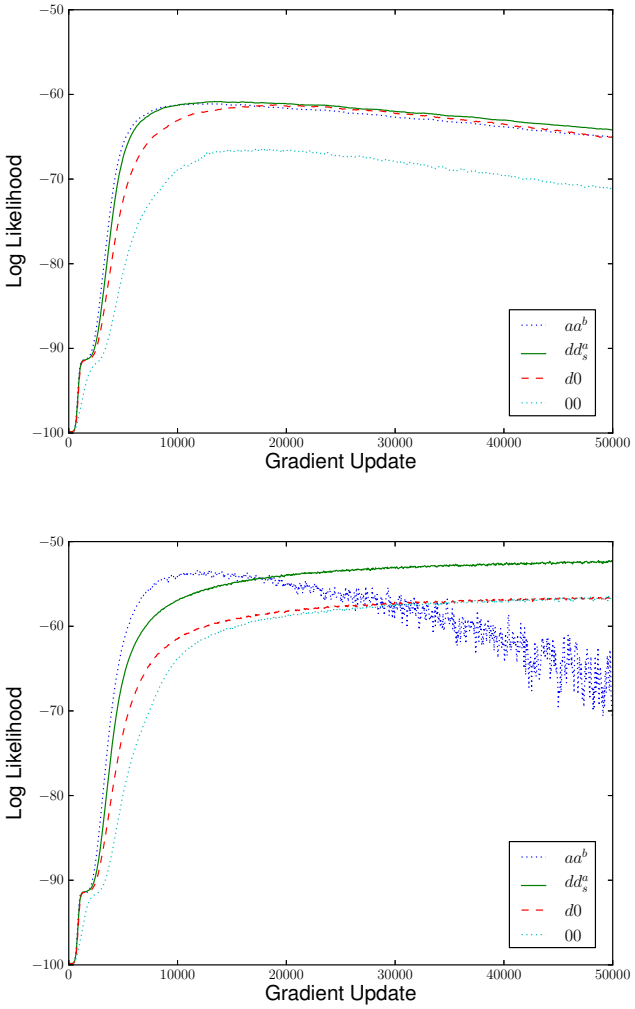

Figure 1: Mean LL during training on the *Bars & Stripes* dataset for the standard methods. Top: CD-1 is used for sampling and the learning rate is $\eta = 0.05$. Bottom: PT_{10} is used for sampling and the learning rate is $\eta = 0.05$.

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

Table 1: Average maximum LL on (top) the *Bars & Stripes* dataset and (bottom) the *MNIST* dataset using different sampling methods and learning rates.

ALGORITHM-η	00 $init\ zero$	00 $init\ sigmoid^{-1}$
CD-1-0.2	-27.98 $\pm$0.26	**-21.49** $\pm$1.34
CD-1-0.1	-28.28 $\pm$0.00	**-21.09** $\pm$0.97
CD-1-0.05	-28.28 $\pm$0.00	**-24.87** $\pm$0.47
PCD-1-0.2	-28.01 $\pm$0.26	**-22.45** $\pm$1.00
PCD-1-0.1	-28.28 $\pm$0.00	**-21.76** $\pm$0.74
PCD-1-0.05	-28.28 $\pm$0.00	**-24.83** $\pm$0.55
PT_{10}-0.2	-28.01 $\pm$0.27	**-21.72** $\pm$1.24
PT_{10}-0.1	-28.28 $\pm$0.00	**-21.14** $\pm$0.85
PT_{10}-0.05	-28.28 $\pm$0.00	**-24.80** $\pm$0.52

Table 2: Average maximum LL on the *Flipped Shifting Bar* dataset, where the visible bias is initialized to zero or to the inverse sigmoid of the data mean.

ALGORITHM-η	$dd_s^a\ init\ zero$	$dd_s^a\ init\ sigmoid^{-1}$
CD-1-0.2	**-20.34** $\pm$0.74	**-20.42** $\pm$0.80
CD-1-0.1	**-20.75** $\pm$0.79	**-20.85** $\pm$0.82
CD-1-0.05	-23.00 $\pm$0.72	**-22.63** $\pm$0.66
PCD-1-0.2	**-21.03** $\pm$0.51	**-20.97** $\pm$0.65
PCD-1-0.1	**-20.86** $\pm$0.75	**-20.72** $\pm$0.50
PCD-1-0.05	-22.75 $\pm$0.66	**-22.30** $\pm$0.64
PT_{10}-0.2	**-20.08** $\pm$0.38	**-20.25** $\pm$0.55
PT_{10}-0.1	**-20.56** $\pm$0.69	**-20.68** $\pm$0.69
PT_{10}-0.05	-22.93 $\pm$0.72	**-22.39** $\pm$0.65

Table 3: Average maximum LL on the *Flipped Shifting Bar* dataset, where the visible bias is initialized to zero or to the inverse sigmoid of the data mean.

5.3 Reparameterization

To analyze the different effects of performing the reparameterization before or after the gradient update we analyzed the learning behavior of dd_s^b and dd_s^a on all datasets. The results for RBMs trained on the *Bars& Stripes* dataset are given in Table 4 (top). No significant difference between both versions can be observed. The same observations can be made for the *Shifting Bar* and *Flipped Shifting Bar* dataset. The results for the *MNIST* dataset are shown in Table 4 (bottom). Here dd_s^b performs slightly better than dd_s^a in the case of CD and no difference could be observed for PCD and PT. We use the reparameterization before the gradient update in the remainder of this work.

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

Table 4: Average maximum LL on (top) the *Bars & Stripes* dataset and (bottom) the *MNIST* dataset, using the reparameterization before and after the gradient update.

5.4 Usage of a sliding average

We analyzed the effect of using a sliding average with a sliding factor of 0.01 for the offset parameters. Interestingly, when training an RBM using PT based on the enhanced gradient a sliding average prevents the observed divergence of the LL. As an example see the learning curves for the *Bars& Stripes* dataset in Figure 2 (top) in comparison to learning curves for training without sliding average Figure 1 (bottom). We still get comparable model performances, however the convergence speed of aa_s^b is reduced. In addition the usage of an sliding average makes the learning curves of the different methods almost equivalent. Note, that dd does not suffer from the divergence problem even when used without sliding average, as can be seen in Figure 2 (bottom) for example. All observation can be made also for the other datasets (see Table 5).

5.5 Other choices for the offsets

As mentioned in section 3, there are other choices for the offset parameters which lead to the same updates for the weights as the enhanced gradient. The choice of $\boldsymbol{\mu} = \langle \mathbf{x} \rangle_d$ and $\boldsymbol{\lambda} = \langle \mathbf{h} \rangle_m$ seems to be reasonable since the data mean is usually known in advanced. We trained an RBM with dm_s^b using a sliding factor of 0.01. The results are shown in Table 5, which suggest that there is no significant difference from aa_s^b

and dd_s^b. However, without a sliding average dm^b has the same divergence problems as aa^b, see Figure 2 (bottom).

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

Table 5: Average maximum LL on (top) *Bars & Stripes*, (middel) *Flipped Shifting Bar* and (bottom) *MNIST* when using a sliding average.

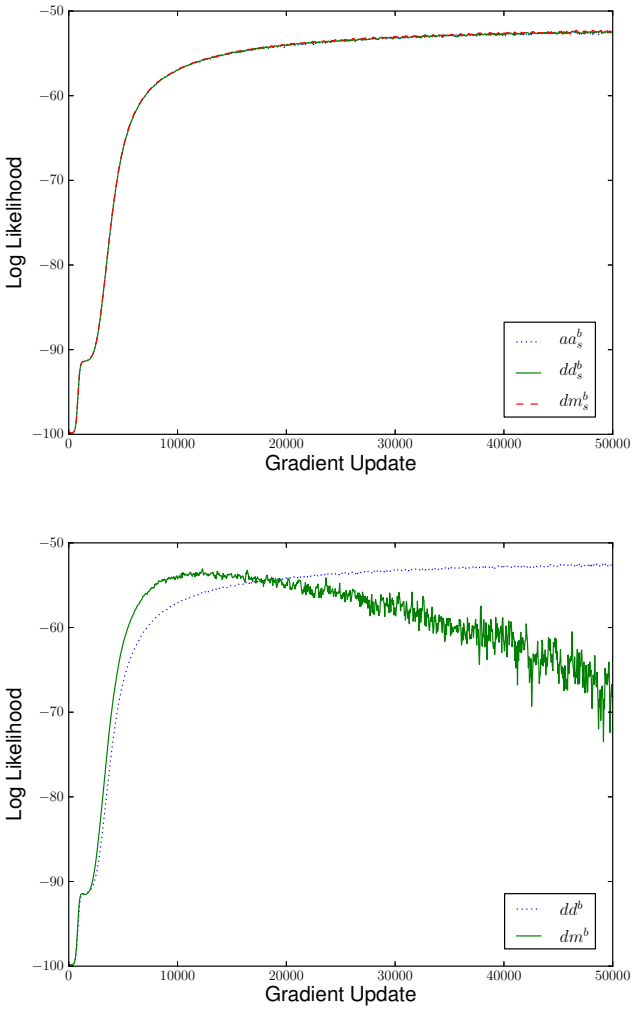

Figure 2: Mean LL during training on the *Bars& Stripes* with the different centering variants using PT_{10}, a learning rate of $\eta = 0.05$, (top) a sliding factor of 0.01 and (bottom) no sliding average.

6 Conclusion

In this paper, we analyze centered RBMs where centering is done by subtracting offset values from visible and hidden variables. The log-likelihood gradient of centered RBMs (and DBMs) is shown to be invariant to variable flips if the corresponding offset parameters flip as well. Training a centered RBM can be reformulated to training a normal binary RBM based on an alternative parameter update. From this new formulation follows that the enhanced gradient is equivalent to centering with a certain choice of offset parameters. Our experiments show that centered RBMs have a better generative performance than normal binary RBMs and thus centered RBMs should be the model of choice. Optimal performance is achieved when centering both, visible and hidden variables, although the improvement is mainly due to centering of the visible variables. A sliding average should be used for the approximations of the offset values, since it stabilizes learning and prevents the severe divergences problems we observed when using the enhanced gradient.

References

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 21(6):1601–1621, 2009.

K. Cho, T. Raiko, and A. Ilin. Enhanced gradient and adaptive learning rate for training restricted boltzmann machines. In Lise Getoor and Tobias Scheffer, editors, *International Conference on Machine Learning (ICML)*, pages 105–112. ACM, 2011.

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

K. Cho, T. Raiko, and A. Ilin. Gaussian-bernoulli deep boltzmann machines. *In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2013). Aug 2013.*, 2013b.

G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Dellaleau. Parallel tempering for training of restricted Boltzmann machines. *JMLR W&CP: AISTATS 2010*, 9: 145–152, 2010.

A. Fischer and C. Igel. Empirical analysis of the divergence of Gibbs sampling based learning algorithms for Restricted Boltzmann Machines. In K. Diamantaras, W. Duch, and L. S. Iliadis, editors, *International Conference on Artificial Neural Networks (ICANN)*, volume 6354 of *LNCS*, pages 208–217. Springer-Verlag, 2010.

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

Y. LeCun, Bottou L., Orr G., and K. Müller. Efficient backprop. *Neural Networks: Tricks of the trade*, 1998.

D. J. C. MacKay. Information theory, inference, and learning algorithm. *Cambridge: Cambridge University Press.*, 2002.

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

H. Schulz, A. Muller, and S. Behnke. Investigating convergence of restricted boltzmann machine learning. *IPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

P. Smolensky. Information processing in dynamical systems: foundations of harmony theory. In *Parallel distributed processing: explorations in the microstructure of cognition*, pages 194–281. MIT Press, Cambridge, MA, USA, 1986.

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 W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *International Conference on Machine learning (ICML)*, pages 1064–1071. ACM, 2008.