LETTER Communicated by Yoshua Bengio

Enhanced Gradient for Training Restricted Boltzmann Machines

KyungHyun Cho
kyunghyun.cho@aalto.fi
Tapani Raiko
tapani.raiko@aalto.fi
Alexander Ilin
alexander.ilin@aalto.fi
Department of Information and Computer Science, Aalto University School of Science, Espoo, Uusimaa 02150, Finland

Restricted Boltzmann machines (RBMs) are often used as building blocks in greedy learning of deep networks. However, training this simple model can be laborious. Traditional learning algorithms often converge only with the right choice of metaparameters that specify, for example, learning rate scheduling and the scale of the initial weights. They are also sensitive to specific data representation. An equivalent RBM can be obtained by flipping some bits and changing the weights and biases accordingly, but traditional learning rules are not invariant to such transformations. Without careful tuning of these training settings, traditional algorithms can easily get stuck or even diverge. In this letter, we present an enhanced gradient that is derived to be invariant to bit-flipping transformations. We experimentally show that the enhanced gradient yields more stable training of RBMs both when used with a fixed learning rate and an adaptive one.

1 Introduction

Deep learning has gained popularity recently as a way for learning complex and large probabilistic models (see, e.g., Bengio, 2009). Especially, deep neural networks such as the deep belief network and the deep Boltzmann machine have been applied to various machine learning tasks with impressive improvements over conventional approaches (Hinton & Salakhutdinov, 2006; Salakhutdinov & Hinton, 2009; Salakhutdinov, 2009; Krizhevsky, 2010; Lee, Grosse, Ranganath, & Ng, 2009).

Deep neural networks are characterized by a large number of layers of neurons and by using layer-wise unsupervised pretraining to learn a probabilistic model for data. A deep neural network is typically constructed by stacking multiple restricted Boltzmann machines (RBM) so that the hidden layer of one RBM becomes the visible layer of another. Layer-wise

pretraining of RBMs then facilitates finding a more accurate model for the data. In cases of performing classification tasks using deep neural networks, various papers (Salakhutdinov & Hinton, 2009; Hinton & Salakhutdinov, 2006; Erhan et al., 2010) have empirically confirmed that such multistage learning works as good as, or in many cases better than, conventional learning methods, such as backpropagation with random initialization. This trend is more apparent when most training samples are unlabeled and only a small number of labeled training samples are available (see, e.g., Ranzato, Huang, Boureau, & LeCun, 2007). It is thus important to have an efficient method for training RBMs.

Unfortunately, training RBMs is known to be difficult. Traditional learning algorithms often converge only with the right choice of metaparameters that specify, for example, learning rate scheduling and the scale of the initial weights.

In this letter, we discuss difficulties of training RBMs using the traditional algorithm and propose a new training algorithm based on a new, enhanced gradient estimate. The new gradient is designed to be invariant to data representation, and it also facilitates learning distinct features by hidden neurons. We show the efficacy of the proposed gradient in experiments with either a fixed or an adaptive learning rate. The preliminary results of this work were presented in our conference paper (Cho, Ilin, & Raiko, 2011) and a technical report (Raiko, Cho, & Ilin, 2011).

2 Restricted Boltzmann Machines

2.1 Model Definition. The restricted Boltzmann machine is a stochastic neural network with a bipartite structure such that each visible neuron is connected to all the hidden neurons and each hidden neuron is connected to all the visible ones (Smolensky, 1986). The energy and the state probability are defined as

$$E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h},$$
$$P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\{-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\}, \tag{2.1}$$

where $\mathbf{v}$ and $\mathbf{h}$ are binary column vectors representing the state of the visible and hidden neurons and parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$ include weights $\mathbf{W} = [w_{ij}]_{n_v \times n_h}$ and biases $\mathbf{b} = [b_i]_{n_v \times 1}$ and $\mathbf{c} = [c_j]_{n_h \times 1}$. n_v and n_h are the numbers of visible and hidden neurons, respectively. $Z(\boldsymbol{\theta})$ denotes the normalizing constant, which is intractable and can be calculated by summing exponentially many terms.

A useful property of the RBM is that hidden neurons $\mathbf{h}$ are independent of each other given visible neurons $\mathbf{v}$,

$$P(h_j = 1 \mid \mathbf{v}, \boldsymbol{\theta}) = \frac{1}{1 + \exp\left(-\sum_i w_{ij} v_i - c_j\right)}, \tag{2.2}$$

and the same holds for the visible neurons:

$$P(v_i = 1 \mid \mathbf{h}, \boldsymbol{\theta}) = \frac{1}{1 + \exp\left(-\sum_j w_{ij} h_j - b_i\right)}. \tag{2.3}$$

This fact allows for efficient parallel implementation of layer-wise Gibbs sampling when collecting samples from the distribution defined by an RBM.

2.2 Training Algorithms. The parameters of an RBM can be learned from data using the standard maximum likelihood estimation. Given a data set $\{\mathbf{v}^{(t)}\}_{t=1}^{N}$, the log likelihood of the parameters is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{t=1}^{N} \log P(\mathbf{v}^{(t)} | \boldsymbol{\theta}) = \sum_{t=1}^{N} \log \sum_{\mathbf{h}} P(\mathbf{v}^{(t)}, \mathbf{h} \mid \boldsymbol{\theta}),$$

where the samples $\mathbf{v}^{(t)}$s are assumed to be independent of each other and the states $\mathbf{h}$ of the hidden neurons are marginalized out.

The gradient ascent update rules are

$$w_{ij} \leftarrow w_{ij} + \eta_w \nabla w_{ij}, \qquad \nabla w_{ij} = \langle v_i h_j \rangle_{\mathrm{d}} - \langle v_i h_j \rangle_{\mathrm{m}}, \tag{2.4}$$

$$b_i \leftarrow b_i + \eta_b \nabla b_i, \qquad \nabla b_i = \langle v_i \rangle_{\mathrm{d}} - \langle v_i \rangle_{\mathrm{m}}, \tag{2.5}$$

$$c_j \leftarrow c_j + \eta_c \nabla c_j, \qquad \nabla c_j = \langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}}, \tag{2.6}$$

where a shorthand notation $\langle \cdot \rangle_{P(\cdot)}$ denotes the expectation computed over probability distribution $P(\cdot)$. $\langle \cdot \rangle_{\mathrm{d}}$ denotes expectation computed over conditional distribution $P(\mathbf{h} \mid \mathbf{v}, \boldsymbol{\theta}) D(\mathbf{v})$ where $D(\mathbf{v})$ is a data distribution and $\langle \cdot \rangle_{\mathrm{m}}$ is expectation computed over the model distribution $P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})$.

The well-known difficulty of using equations 2.4 to 2.6 is that while the expectations $\langle \cdot \rangle_{\mathrm{d}}$ can easily be calculated using equation 2.2, the exact computation of expectations $\langle \cdot \rangle_{\mathrm{m}}$ is intractable because of the need to compute the normalizing constant.

Conventional learning procedures employ the stochastic gradient ascent method, which uses only a small subset of training data samples, called a minibatch, to compute the gradients at each update.

2.2.1 Contrastive Divergence. An efficient method for training RBMs is based on minimizing contrastive divergence (CD; Hinton, 2002). In this approach, the true gradients in equation 2.4 to 2.6 are approximated by replacing expectations $\langle\cdot\rangle_{\mathrm{m}}$ with expectations $\langle\cdot\rangle_{P_n}$ evaluated over the distribution P_n obtained by running n steps of the layer-wise Gibbs sampling, starting from the empirical distribution defined by the training samples. This yields the following update rule for the weights:

$$w_{ij} \leftarrow w_{ij} + \eta_w\left[\langle x_i h_j\rangle_{P_0} - \langle x_i h_j\rangle_{P_n}\right],$$

where P_0 denotes distribution $P(\mathbf{h} \mid \mathbf{v}, \boldsymbol{\theta})$ with $\mathbf{v}$ fixed to the training data. In practice, using a short Gibbs sampling chain (e.g., $n = 1$) often yields good performance.

2.2.2 Approximate Maximum Likelihood. Minimizing CD is known to be biased for finite n and to provide the maximum likelihood solution only when $n \to \infty$ (Carreira-Perpiñán & Hinton, 2005; Bengio & Delalleau, 2009). This problem is fixed in methods that use the stochastic approximation to the likelihood gradient (Younes, 1989), which yields a different procedure for collecting samples from the model distribution: The main difference in implementation compared to CD is that sampling is not started at the training samples for each minibatch. Existing approaches include persistent contrastive divergence (PCD; Tieleman, 2008), tempered transition (Salakhutdinov, 2009), and parallel tempering (PT; Desjardins, Courville, Bengio, Vincent, & Delalleau, 2010; Cho, Raiko, & Ilin, 2010).

In this letter, we use PCD and PT as representative methods of this kind. PCD is based on plain Gibbs sampling. The basic idea of PT is that multiple chains of Gibbs sampling are run for models with different "temperatures." Chains with higher temperatures correspond to more diffuse distributions and therefore can produce a greater variety of samples. Every now and then, the samples are swapped between the chains, which facilitates better exploration of the state space.

2.3 Difficulties in Training RBM. Training RBMs can be difficult in practice. Due to the intractability of the objective function, it is difficult to compare the quality of found solutions or even to know when learning has converged. It has been observed that the training procedure can diverge especially when Gibbs sampling is used to obtain samples from the model distribution (Desjardins, Courville, Bengio, Vincent et al., 2010; Schulz, Müller, & Behnke, 2010; Fischer & Igel, 2010). This diverging behavior

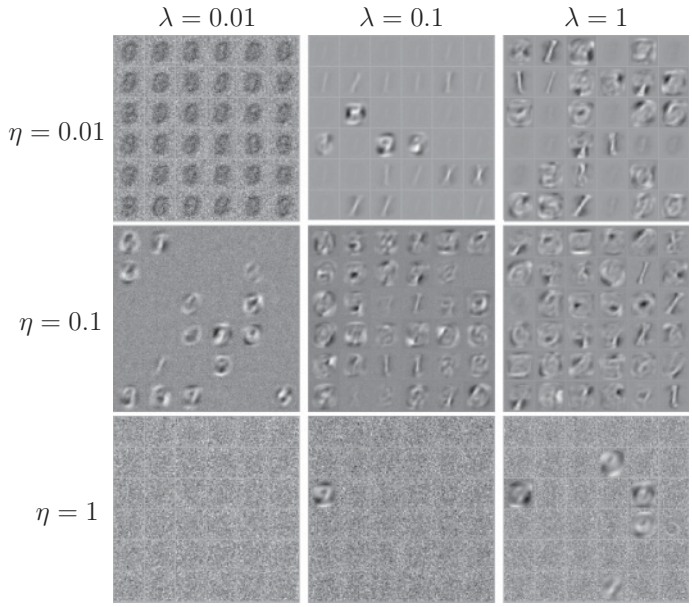


Figure 1: Visualization of the filters learned by RBMs on MNIST with various learning rates η and initial weight scaling λ. Learning was performed for five epochs, each using the traditional gradient with a fixed learning rate.

can be suppressed by using more sophisticated sampling methods, but the likelihood can still fluctuate highly during training unless one uses appropriate learning rate scheduling or adapts the sampler to maintain a certain level of mixing property (Desjardins, Courville, & Bengio, 2010; Desjardins, Courville, Bengio, Vincent et al., 2010; Desjardins, Courville, Bengio, Vincent, & Delalleau, 2009).

2.3.1 Sensitivity to Metaparameters. In Figure 1, we demonstrate the sensitivity of the training procedure based on the traditional gradient in equations 2.4 to 2.6 to the scale of weight initialization and the learning rate. In this experiment, RBMs with 36 hidden neurons were trained on the MNIST data set (LeCun, Bottou, Bengio, & Haffner, 1998) with fixed learning rates η and PT as the sampling strategy to obtain samples from the model distribution. Weights w_{ij} were initialized with random values drawn from the normal distribution with zero mean and standard deviation λ. Each visible bias b_i was initialized to $\log \frac{m_i}{1-m_i}$, where m_i is the sample mean of the ith pixel in the training data, and all hidden biases c_j were initially set to -4. The figure presents the filters learned by RBMs for different combinations of λ and η.

It is clear that the results after a relatively short training period are highly dependent on the choice of metaparameters. Reasonable features are learned by most of the hidden neurons only with a careful selection of the initial weight scale and the learning rate, which was $\eta = 0.1$ and $\lambda = 1$ for the reported experiments. In longer runs, training results generally improve, and the difference between different training scenarios may be less dramatic. Nevertheless, this result suggests that careful selection of the metaparameters is very important for RBMs. The optimal combination of metaparameters is usually found by cross-validation, which may become time-consuming when there are many metaparameters to tune (see, e.g., Bergstra & Bengio, 2012). Hence, it is preferable to have a learning algorithm that is less sensitive to metaparameters.

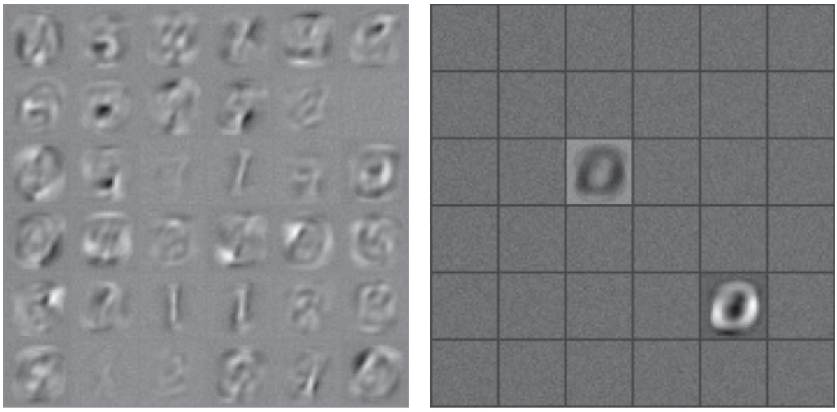
The results obtained with $\eta = 0.01$ illustrate a typical problem in training RBMs when several hidden neurons try to learn global filters that resemble the visible bias term $\mathbf{b}$ (see, e.g., Hinton, 2010). Such hidden neurons are activated for most of the input objects, but they are meaningless because the corresponding weights can be incorporated into biases b_i. One can also observe noisy global filters that do not seem to capture any meaningful features, especially in the results obtained with large η. Such hidden neurons are always inactive and can be removed without affecting the modeling capacity of the RBM. The existence of such hidden neurons is an indicator of poorly trained RBMs.

2.3.2 Sensitivity to Data Representation. In Figure 2, we demonstrate that the training procedure based on the traditional gradient in equations 2.4 to 2.6 is also sensitive to data representation. For example, flipping all the bits in the MNIST data set (such that zeros become ones and vice versa) produces an equivalent data set, which we call 1-MNIST. However, training results obtained on MNIST and 1-MNIST can be very different. The RBM trained on 1-MNIST after a relatively short training period does not contain any meaningful features: two hidden neurons model something similar to the visible bias, and the remaining ones are always inactive. In contrast, the RBM trained on MNIST has learned several reasonable features. It suggests that the learning algorithm based on the traditional gradient update is highly sensitive to the data representation.

3 Enhanced Gradient

3.1 Derivation of the Enhanced Gradient. In this section, we propose a new gradient to modify the update rules 2.4 to 2.6. We first define the covariance between two variables under distribution P as

$$\mathrm{Cov}_P(v_i, h_j) = \langle v_i h_j \rangle_P - \langle v_i \rangle_P \langle h_j \rangle_P.$$



(a) MNIST (b) 1-MNIST

Figure 2: Visualization of the filters learned by RBMs with 36 hidden neurons on MNIST and 1-MNIST using PT sampling after five epochs. Both RBMs were trained with the traditional learning rate and a fixed learning rate 0.1. The initial weights were sampled from the normal distribution with zero mean and standard deviation 0.1.

Then the standard gradient in equation 2.4 can be rewritten as

$$\nabla w_{ij} = \mathrm{Cov}_{\mathrm{d}}(v_i, h_j) - \mathrm{Cov}_{\mathrm{m}}(v_i, h_j) + \langle v_i \rangle_{\mathrm{dm}} \nabla c_j + \langle h_j \rangle_{\mathrm{dm}} \nabla b_i, \tag{3.1}$$

where ∇c_j and ∇b_i are the gradients that appear in equations 2.5 and 2.6 and $\langle \cdot \rangle_{\mathrm{dm}} = \frac{1}{2}\langle \cdot \rangle_{\mathrm{d}} + \frac{1}{2}\langle \cdot \rangle_{\mathrm{m}}$ is the average activity of a neuron under the data and model distributions.

One potential problem with the gradient 3.1 is that it contains terms ∇c_j, ∇b_i that point in the same direction as the gradient with respect to the bias terms. This effect is prominent when there are many neurons that are mainly active, that is, for which $\langle x_k \rangle_{\mathrm{dm}} \approx 1$, where x_k can be either a visible or a hidden neuron. These terms can distract learning of meaningful weights by making many neurons learn features resembling the bias terms, the effect that is visible in Figures 1 and 2. When $\langle x_i \rangle_{\mathrm{dm}} \approx 0$ for most of the neurons, this effect can be negligible, which might explain why learning 1-MNIST is more difficult than MNIST and partially explain why sparse Boltzmann machines (Lee, Ekanadham, & Ng, 2008) have been successful.

Another problem of using gradient 3.1 is that the updated parameter values are different depending on the data representation. This can be

shown by using transformations where some of the binary units are flipped such that zeros become ones and vice versa:

$$\tilde{x}_k = x_k^{1-f_k}(1-x_k)^{f_k}, \qquad f_k \in \{0,1\}.$$

The parameters can then be transformed accordingly to $\tilde{\boldsymbol{\theta}}$:

$$\tilde{w}_{ij} = (-1)^{f_i+f_j} w_{ij}, \tag{3.2}$$

$$\tilde{b}_i = (-1)^{f_i}\left(b_i + \sum_j f_j w_{ij}\right), \tag{3.3}$$

$$\tilde{c}_j = (-1)^{f_j}\left(c_j + \sum_i f_i w_{ij}\right), \tag{3.4}$$

such that the resulting RBM has an equivalent energy function, that is, $E(\tilde{\mathbf{x}} \mid \tilde{\boldsymbol{\theta}}) = E(\mathbf{x} \mid \boldsymbol{\theta}) + \text{const}$ for all $\mathbf{x}$ (see the proof in appendix A).

When a model is transformed, updated, and transformed back, the resulting model depends on the transformations f_k:

$$\begin{aligned} w_{ij} &\leftarrow w_{ij} + \eta[\langle v_i h_j\rangle_{\mathrm{d}} - \langle v_i h_j\rangle_{\mathrm{m}} - f_i(\langle h_j\rangle_{\mathrm{d}} - \langle h_j\rangle_{\mathrm{m}}) - f_j(\langle v_i\rangle_{\mathrm{d}} - \langle v_i\rangle_{\mathrm{m}})] \\ &= w_{ij} + \eta[\mathrm{Cov}_{\mathrm{d}}(v_i, h_j) - \mathrm{Cov}_{\mathrm{m}}(v_i, h_j) \\ &\quad + (\langle v_i\rangle_{\mathrm{dm}} - f_i)\nabla c_j + (\langle h_j\rangle_{\mathrm{dm}} - f_j)\nabla b_i] \\ b_i &\leftarrow b_i + \eta\left[\nabla b_i - \sum_j f_j(\nabla w_{ij} - f_i \nabla c_j - f_j \nabla b_i)\right] \\ c_j &\leftarrow c_j + \eta\left[\nabla c_j - \sum_i f_i(\nabla w_{ij} - f_i \nabla c_j - f_j \nabla b_i)\right], \end{aligned}$$

where $\nabla\boldsymbol{\theta}$ are the gradients used in equations 2.4 to 2.6 and we assume that the learning rates for weights are biases are same. This is shown in appendix B.

Thus, there are $2^{n_v+n_h}$ different update rules defined by different combinations of binary f_k, $k = 1, \ldots, n_v + n_h$. All the update rules are well-founded maximum likelihood updates to the original model.

The new gradient is, then, proposed to be a weighted sum of the $2^{n_v+n_h}$ gradients with the following weights:

$$\prod_{k=1}^{n_v+n_h} \langle x_k\rangle_{\mathrm{dm}}^{f_k}\left(1 - \langle x_k\rangle_{\mathrm{dm}}\right)^{1-f_k}. \tag{3.5}$$

By using these weights, the new gradient prefers sparse data representations for which $\langle x_k \rangle_{\mathrm{dm}} \approx 0$ because the corresponding models get larger weights.

The proposed weighted sum yields the enhanced gradient (see appendix C),

$$\nabla_e w_{ij} = \mathrm{Cov}_{\mathrm{d}}(v_i, h_j) - \mathrm{Cov}_{\mathrm{m}}(v_i, h_j), \tag{3.6}$$

$$\nabla_e b_i = \langle v_i \rangle_{\mathrm{d}} - \langle v_i \rangle_{\mathrm{m}} - \sum_j \langle h_j \rangle_{\mathrm{dm}} \nabla_e w_{ij}, \tag{3.7}$$

$$\nabla_e c_j = \langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}} - \sum_i \langle v_i \rangle_{\mathrm{dm}} \nabla_e w_{ij}, \tag{3.8}$$

in which, by the choice of the weights 3.5, the effect of the bias gradient terms in the representation 3.1 is canceled out. Thus, the new update rules are

$$w_{ij} \leftarrow w_{ij} + \eta \nabla_e w_{ij}, \tag{3.9}$$

$$b_i \leftarrow b_i + \eta \nabla_e b_i, \tag{3.10}$$

$$c_j \leftarrow c_j + \eta \nabla_e c_j. \tag{3.11}$$

3.2 Analysis of the Enhanced Gradient. In appendix D, we show that the new update rules are invariant to the bit-flipping transformations. It is also easy to see that the enhanced gradient shares all zeros with the traditional gradient.

We compare the enhanced gradient to the traditional one on an RBM with 361 hidden neurons trained on MNIST. Figure 3 represents the angles between the update directions for the hidden neurons. Each element of the visualized matrices is the absolute value of the cosine of the angle between the update directions for two neurons. The gradients obtained with the traditional rule are highly correlated with each other. On the contrary, the new gradient yields update directions that are close to orthogonal, which allows the neurons to learn distinct features.

More details on how to obtain the bit-flipping transformation, the transformation-dependent update rules, and the enhanced gradient update rules for general Boltzmann machines are presented in our technical report (Raiko et al., 2011).

4 Experiments

In this section, we experimentally compare the proposed enhanced gradient with the traditional one.

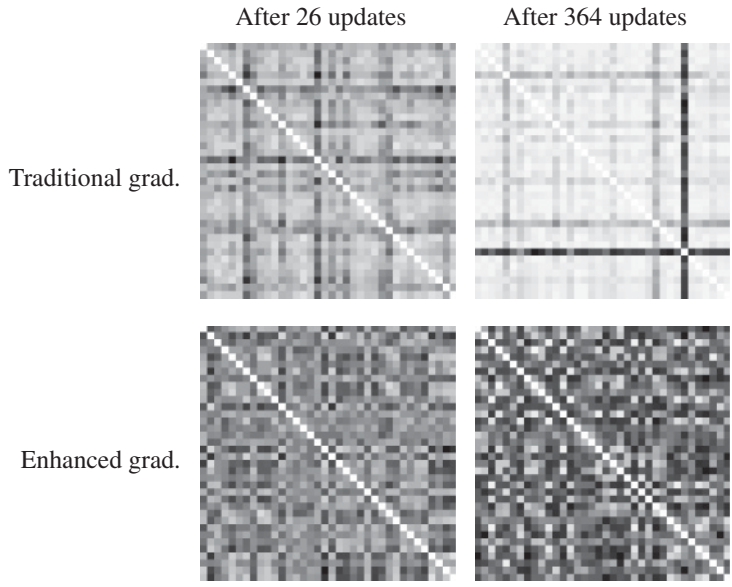


Figure 3: The angles between the update directions for the weights of the RBM with 361 hidden neurons. White pixels correspond to small angles, and black pixels correspond to orthogonal directions.

4.1 Experiments with MNIST and 1-MNIST

4.1.1 Experiments with a Relatively Small Model. We start by running the same experiment as in section 2.3.1 to test the sensitivity of the training procedure based on the enhanced gradient to the scale initialization and the learning rate. As can be seen from Figure 4, the results obtained with the enhanced gradient look better than the ones obtained with the traditional gradient (compare to Figure 1). After a relatively short training period, there are generally more filters that represent some meaningful features, which suggests that using the enhanced gradient can speed up learning.

In the next experiment, we compare the quality of resulting RBMs in longer training sessions. In order to be able to test different training settings in a reasonable amount of time, we train a relatively small RBM with only 361 hidden neurons. We trained RBMs on binarized MNIST and 1-MNIST data sets, in which the original grayscale pixels were rounded. RBMs were initialized according to the practical guide by Hinton (2010): The weights were randomly sampled from the normal distribution with zero mean and standard deviation $\frac{1}{n_v+n_h}$. Each visible bias b_i was initialized to $\log \frac{m_i}{1-m_i}$, where m_i is the sample mean of the ith pixel in the training data. All hidden biases c_j were initially set to -4.

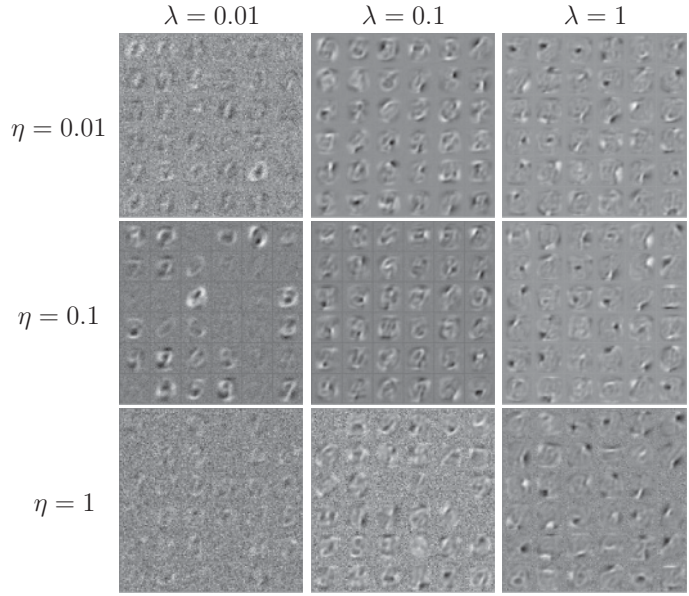


Figure 4: Visualization of the filters learned by RBMs with 36 hidden neurons on MNIST with various initial learning rates η and initial weight scaling λ. Learning was performed for five epochs each using the enhanced gradient with a fixed learning rate.

We ran a set of training sessions for 20 epochs each, varying the learning rate and the strategy for collecting samples from the model distribution. We used PT with 11 equally spaced temperatures and PCD and CD with a single Gibbs update ($n = 1$). The learning rate was initialized with values 2^l, where l was randomly sampled from a uniform distribution $[-9, 3]$. We tried both fixing the learning rate and using the adaptation scheme based on maximizing a local approximation of the likelihood (Cho, Ilin et al., 2011).

The quality of the trained models is assessed using the following quantities:

1. Log probability of test data under the trained model. The computation of this quantity requires the knowledge of the normalizing constant, which was estimated using annealed importance sampling (AIS), similarly to previous work (Salakhutdinov, 2009). The estimates of the normalizing constant were obtained by averaging 100 independent runs of AIS with different initializations. Each AIS run used 10,001 equally spaced temperatures.
2. Classification accuracy obtained with a logistic regression classifier trained on the activation probabilities of the hidden neurons

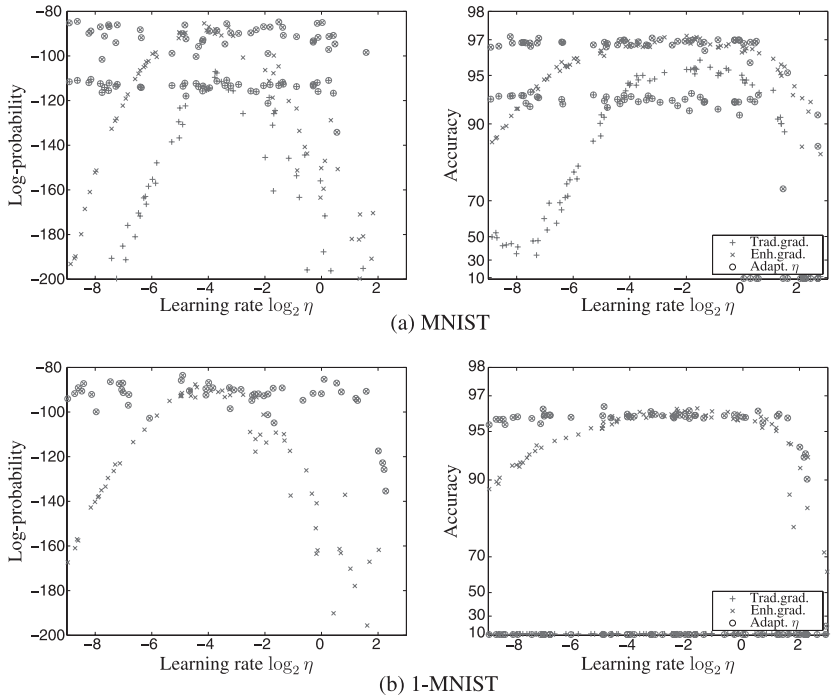


Figure 5: Log probabilities and classification accuracies of test data for different initializations of the learning rate. The models were trained using the stochastic approximation with PT sampling.

conditioned on the original grayscale pixels. Additionally, we fine-tuned the network using stochastic backpropagation. Note that these are indirect measures of the RBM quality because they are not optimized during learning.

Figure 5 shows the comparison of the training results obtained with PT. Each marker represents a value of a performance index against the initial value of the learning rate. A cross corresponds to the traditional gradient, while an $\times$ corresponds to the enhanced gradient. Markers surrounded by circles represent the results obtained with the adaptive learning rate.

The main observation from Figure 5 is that in the case of PT, training with the enhanced gradient generally results in higher log probabilities and better classification accuracies. The variation of the results is much higher for the traditional approach, while most of the runs with the enhanced gradient provided relatively good RBMs. The improvement of the RBM quality is significant for MNIST and radical for 1-MNIST, where we were not able to train any reasonable model with the traditional gradient. Note

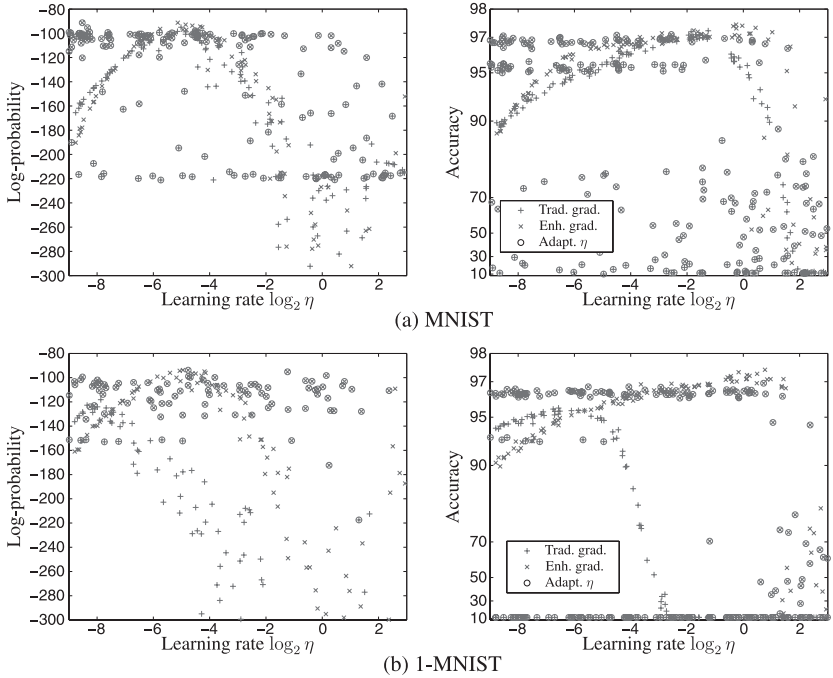


Figure 6: Log probabilities and classification accuracies of test data for different initializations of the learning rate. The models were trained using the stochastic approximation with Gibbs sampling. Note the difference in the y-axis scaling from Figures 5 and 7.

also that using the adaptive learning rate led in most cases to a relatively good model regardless of the learning rate initialization when it was used with the enhanced gradient. However, the adaptation mechanism did not find the optimal learning rate in the case of the traditional gradient, and the best results were obtained with a learning rate fixed to a proper value.

In spite of the fact that the enhanced gradient is invariant to data representation, note that the classification accuracies obtained on 1-MNIST are slightly worse than those on MNIST. This is due to different initial sparsity of the hidden units h_j. Approximately equivalent results for 1-MNIST and MNIST could be obtained when the hidden biases were initialized to $c_j \leftarrow -4 - \sum_i w_{ij} \langle v_i \rangle_d$. However, for consistency, we present only the results obtained with the initialization procedure proposed by Hinton (2010).

The results obtained using PCD with Gibbs sampling are shown in Figure 6. They indicate that the enhanced gradient is again superior to the traditional gradient. It is especially apparent in the case of 1-MNIST, where the enhanced gradient shows more robustness to changes in the

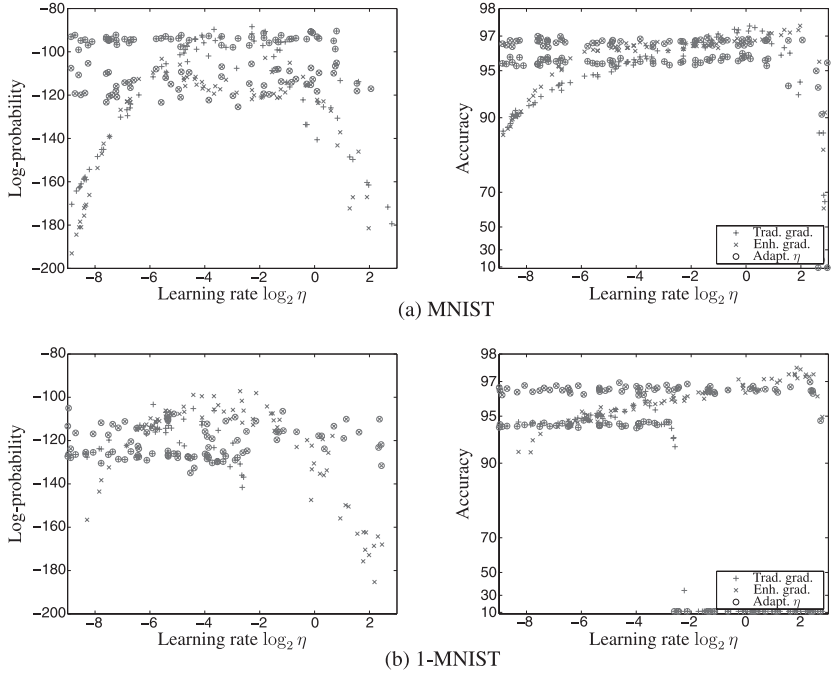
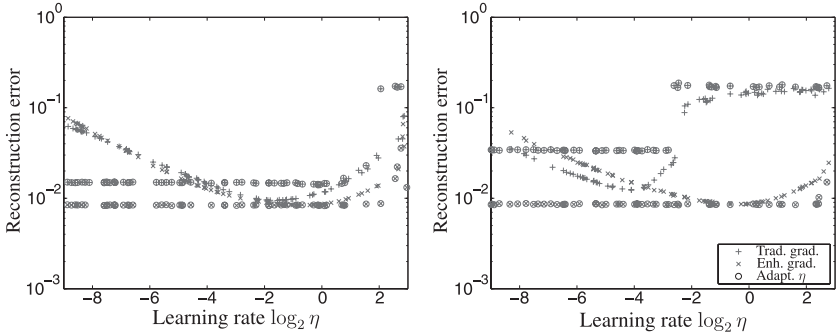


Figure 7: Log probabilities and classification accuracies of test data for different initializations of the learning rate. The models were trained by minimizing CD.

learning rate. In a few cases, we can observe that the adaptive learning rate was not able to adapt the learning rate appropriately. However, the results suggest that the enhanced gradient, together with the adaptive learning rate, performs better compared to the traditional gradient with or without the adaptive learning rate.

Figure 7 shows the results of the same experiment with CD. Again, learning from 1-MNIST is much easier with the enhanced gradient; note that learning with the traditional gradient often failed completely. The best classification accuracies obtained with the two gradients were quite similar. However, using the traditional gradient resulted in better generative models for MNIST. A possible explanation is that learning by minimizing CD does not maximize the log likelihood directly, and therefore one should not regard log likelihood as the ultimate performance indicator. Especially using a small number of Gibbs steps introduces a large bias in favor of minimizing the one-step reconstruction error (Bengio & Delalleau, 2009).

In Figure 8, we show the average mean-squared reconstruction error of the test samples. It is clear that in most cases, the models trained with the enhanced gradient achieved lower reconstruction errors. Together with the



(a) MNIST (b) 1-MNIST

Figure 8: Mean-squared one-step reconstruction errors of test samples for the models trained by minimizing CD. Note that smaller reconstruction error values represent better models, unlike log probabilities.

adaptive learning, those models were consistently better than the models trained with the traditional gradient for a broad range of initial learning rates.

4.1.2 Experiments with Larger Models. Additionally, in order to get higher classification accuracies on MNIST, we trained larger RBMs having 500, 1000, 2000, and 4000 hidden neurons. In this experiment, we used PT with 11 temperatures and CD with a single Gibbs step. All the models were trained for 1000 epochs, with the adaptive learning rate initialized to 0.0001 and upper-bounded by 0.1. In the second half of training, the learning rate was annealed inverse proportionally to the number of updates. In the case of PT, we also varied the number of model samples collected to compute the expectation over the model distribution. We used 128 and 1024 samples, which is marked by PT-128 and PT-1024 in the experimental results.

The achieved classification accuracies are shown in Table 1. The best result of 98.49% was obtained with an RBM having 4000 hidden neurons, which was trained with the enhanced gradient by minimizing CD after fine-tuning the model. In the case of using PT, we observed that the models trained with the enhanced gradient outperformed those trained with the traditional gradient. However, the accuracies were lower than those obtained by the models trained by minimizing CD.

The RBM having 4000 hidden neurons, trained by computing the expectation over the model distribution with 1024 samples collected by PT sampling, achieved the highest log probability, -64.06. However, the models trained with 128 model samples were not able to perform as well as those with 1024 model samples. Furthermore, the RBM trained by minimizing CD with the enhanced gradient systematically had lower one-step

Table 1: Classification Accuracies of the Test Data of MNIST.

	Before Fine-Tuning				After Fine-Tuning			
	PT (128)		CD		PT (128)		CD	
Hidden Neurons	Enhanced	Traditional	Enhanced	Traditional	Enhanced	Traditional	Enhanced	Traditional
500	97.58	97.25	97.15	97.04	97.58	97.18	97.15	97.04
1000	97.93	97.78	98.02	97.82	97.94	97.78	98.09	97.83
2000	98.33	97.89	98.17	97.81	98.31	97.94	98.16	97.80
4000	98.41	97.82	98.48	98.11	98.36	97.83	**98.49**	98.10

Note: The highest accuracy is shown in boldface.

reconstruction errors of the test samples compared to those trained with the traditional gradient.

One noticeable phenomenon observed when the RBMs having a large number of hidden neurons, such as 2000 or 4000, were trained, was that the enhanced gradient used more hidden neurons than the traditional gradient did. It can be investigated by computing $\mathrm{Var}[p(h_j \mid \mathbf{v})]$, where $\mathbf{v}$ comes from test data. Those unused hidden neurons will have a variance close to 0, indicating that they are not dependent on whatever samples are clamped to the visible neurons. In Figure 9, it is clear that the models trained with the enhanced gradient have more hidden neurons with nonzero variances of the conditional probabilities.

4.2 Experiments with Caltech 101 Silhouettes. In this section, we test the proposed enhanced gradient on the Caltech 101 Silhouettes data (Marlin, Swersky, Chen, & de Freitas, 2010; random samples from this data set are shown in Figure 10). We ran four experiments with RBMs with 500, 1000, 2000, and 4000 hidden neurons. In order to avoid tuning the learning rate, we used the adaptive learning rate. Training was performed for 1000 epochs with PT and the minibatch size 128. The learning rate was initialized to 0.0001, and the upper bound was set to 0.1. After 500 epochs, the learning rate was decreased inverse proportionally to the number of updates.

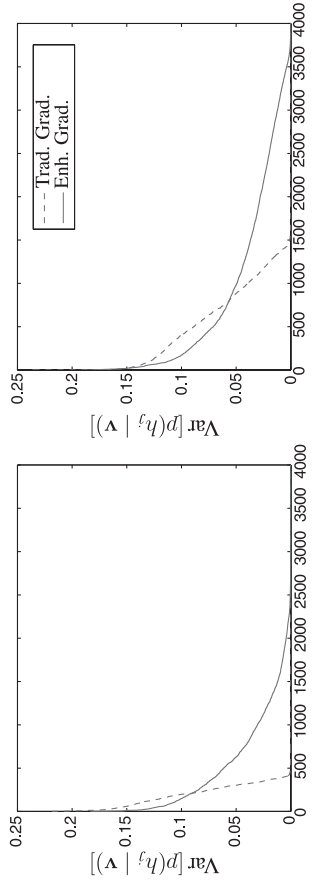
The obtained results are presented in Tables 2 to 5, where we used the same performance indexes as in section 4.1 to assess the quality of the results. Figure 10b shows samples drawn from the trained model. Remarkably, the classification accuracy improved by more than 5% over the best result reported by Marlin et al. (2010), with higher log probabilities when the enhanced gradient was used. The trend is more evident when PT was used to sample from the model distributions. However, we were not able to improve the accuracies with the discriminative fine-tuning.

We emphasize that we used the enhanced gradient with the adaptive learning rate without laborious tuning. In contrast, Marlin et al. (2010) used an extensive validation to choose the right learning rate and other metaparameters and also applied an advanced optimization method as a fine-tuning method to the initial stochastic gradient descent optimization.

5 Conclusion

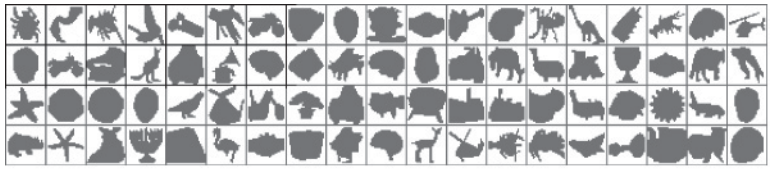
In this letter, we discussed several potential problems with training RBMs and proposed new update rules, based on the enhanced gradient. Unlike the traditional learning rules, which are dependent on data representation, the enhanced gradient was derived to be invariant to it. This allowed learning the flipped version of the MNIST data set without any difficulty.

The proposed gradient was compared against the traditional one using contrastive divergence, parallel tempering, and persistent contrastive divergence with plain Gibbs sampling. For MNIST, the classification accuracies obtained with the enhanced gradient were similar to the traditional one

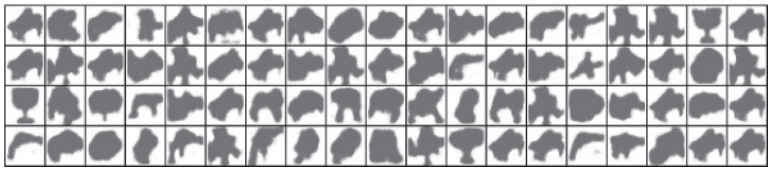


(a) PT (128) (b) PT (1024) (c) CD

Figure 9: Variances of hidden activation probabilities of an RBM having 4000 hidden neurons given test samples of MNIST. The x-axis corresponds to each hidden neuron, and the y-axis represents the variance of the activation probabilities of the neuron given test samples. Hidden units are sorted by the variance of their conditional probabilities.



(a) Caltech 101 Silhouettes



(b) Generated samples

Figure 10: (a) One hundred randomly chosen training samples of Caltech 101 silhouettes. (b) Samples generated from the RBM with 2000 hidden neurons trained on Caltech 101 silhouettes.

Table 2: Log Probabilities of the Test Data of Caltech 101 Silhouettes by the RBMs Trained Using PT.

	PT		
Hidden Neurons	Enhanced	Traditional	(M)
500	−114.75	−179.09	≈−120
1000	−111.72	−184.42	
2000	−108.98	−171.03	
4000	−107.78	−129.04	

Note: (M): Results reported by Marlin et al. (2010) using PCD.

Table 3: Log Probabilities and Reconstruction Errors of the Test Data of Caltech 101 Silhouettes by the RBMs Trained by Minimizing CD.

	Log Probability			Reconstruction Error	
Hidden Neurons	Enhanced	Traditional	(M)	Enhanced	Traditional
500	−241.46	−270.63	<−450	0.0166	0.2447
1000	−228.86	−252.54		0.0117	0.2362
2000	−220.30	−132.15		0.0544	0.2481
4000	−196.36	−176.28		0.0588	0.2489

Note: (M): Result reported by Marlin et al. (2010) using CD.

Table 4: Classification Accuracy of the Test Data of Caltech 101 Silhouettes.

	PT			CD		
Hidden Neurons	Enhanced	Traditional	(M)	Enhanced	Traditional	(M)
500	70.91	67.92	65.4	69.14	70.22	64.9
1000	72.22	67.10		70.61	70.83	
2000	72.17	69.57		72.52	71.04	
4000	73.00	71.26		72.35	69.09	

Note: (M): Results reported by Marlin et al. (2010) using PCD (left) and CD (right).

Table 5: Classification Accuracy of the Test Data of Caltech 101 Silhouettes After Discriminative Fine-Tuning.

	PT			CD		
Hidden Neurons	Enhanced	Traditional	(M)	Enhanced	Traditional	(M)
500	70.83	67.75	65.4	68.96	70.05	64.9
1000	72.17	67.45		70.57	70.74	
2000	72.17	69.14		72.61	71.11	
4000	72.82	71.26		72.39	69.09	

Note: (M): Results reported by Marlin et al. (2010) using PCD (left) and CD (right).

in the case of CD and significantly better in the case of PT and PCD. For 1-MNIST, the enhanced gradient clearly outperformed the traditional one. Our experiments also showed that the enhanced gradient is less sensitive to the choice of metaparameters than the traditional one, which makes training RBMs easier.

Finally, we note that the idea of the enhanced gradient can be applied to other types of Boltzmann machines. For example, we showed some preliminary experimental results on applying this idea to Boltzmann machines with gaussian visible neurons (Cho, Ilin et al., 2011; Cho, Raiko, & Ilin, 2011). Using the enhanced gradient for training other types of Boltzmann machines is a subject for future research.

Appendix A: Bit-Flip Transformations

In this section, we present and prove two properties of the transformations 3.2 to 3.4.

Theorem 1. *The transformed machine is equivalent to the original one, that is,* $P(\tilde{\boldsymbol{x}} \mid \tilde{\boldsymbol{\theta}}) = P(\boldsymbol{x} \mid \boldsymbol{\theta})$ *for all states* $\boldsymbol{x}$.

Proof.

$$
\begin{aligned}
E(\tilde{\mathbf{x}} \mid \tilde{\boldsymbol{\theta}}) &= -\frac{1}{2}\tilde{\mathbf{x}}^{\top}\tilde{\mathbf{W}}\tilde{\mathbf{x}} - \tilde{\mathbf{b}}^{\top}\tilde{\mathbf{x}} \\
&= \sum_{ij} -\frac{1}{2} x_i^{1-f_i}(1-x_i)^{f_i}(-1)^{f_i+f_j} w_{ij} x_j^{1-f_j}(1-x_j)^{f_j} \\
&\quad - \sum_i (-1)^{f_i}\left(b_i + \sum_j f_j w_{ij}\right) x_i^{1-f_i}(1-x_i)^{f_i} \\
&= \sum_{ij} \frac{1}{2}(x_i - f_i) w_{ij}(x_j - f_j) - \sum_i \left(b_i + \sum_j f_j w_{ij}\right)(x_i - f_i) \\
&= \sum_{ij} -\frac{1}{2} x_i w_{ij} x_j - \sum_i b_i x_i + \sum_{ij} \frac{1}{2} x_i w_{ij} f_j + \sum_{ij} \frac{1}{2} x_i W_{ij} x_j \\
&\quad - \sum_{ij} \frac{1}{2} f_i w_{ij} f_j - \sum_{ij} f_j w_{ij} x_i + \sum_i \left(b_i + \sum_j f_j w_{ij}\right) f_i \\
&= \sum_{ij} -\frac{1}{2} x_i w_{ij} x_j - \sum_i b_i x_i \\
&\quad - \sum_{ij} \frac{1}{2} f_i w_{ij} f_j + \sum_i \left(b_i + \sum_j f_j w_{ij}\right) f_i \\
&= E(\mathbf{x} \mid \boldsymbol{\theta}) + \text{const.}
\end{aligned}
$$

Since the energy functions are the same up to a constant, the probability distributions P are the same.

Appendix B: Transformation-Dependent Update Rules

Here we derive and show that the traditional gradient update rules are dependent on the bit-flipping transformation.

We define

$$
\begin{aligned}
\mathrm{Cov}_P(x_i, x_j) &= \langle x_i x_j \rangle_P - \langle x_i \rangle_P \langle x_j \rangle_P \\
\langle \cdot \rangle_{\mathrm{dm}} &= \frac{\langle \cdot \rangle_{\mathrm{d}} + \langle \cdot \rangle_{\mathrm{m}}}{2}
\end{aligned}
$$

and note that

$$\langle x_i\rangle_{\mathrm{dm}}\nabla b_j = \frac{1}{2}\langle x_i\rangle_{\mathrm{d}}\langle x_j\rangle_{\mathrm{d}} + \frac{1}{2}\langle x_i\rangle_{\mathrm{m}}\langle x_j\rangle_{\mathrm{d}} - \frac{1}{2}\langle x_i\rangle_{\mathrm{d}}\langle x_j\rangle_{\mathrm{m}} - \frac{1}{2}\langle x_i\rangle_{\mathrm{m}}\langle x_j\rangle_{\mathrm{m}}$$

to help get the form in equation B.1. The three-step update for the weights obtained by transforming, updating, and transforming back is

$$\begin{aligned}
w_{ij} &\leftarrow (-1)^{f_i+f_j}\left[(-1)^{f_i+f_j}w_{ij} + \eta(\langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{d}} - \langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{m}})\right] \\
&= w_{ij} + \eta(-1)^{f_i+f_j}\left[\left\langle x_i^{1-f_i}(1-x_i)^{f_i}x_j^{1-f_j}(1-x_j)^{f_j}\right\rangle_{\mathrm{d}}\right. \\
&\quad \left.-\left\langle x_i^{1-f_i}(1-x_i)^{f_i}x_j^{1-f_j}(1-x_j)^{f_j}\right\rangle_{\mathrm{m}}\right] \\
&= w_{ij} + \eta[\langle(x_i-f_i)(x_j-f_j)\rangle_{\mathrm{d}} - \langle(x_i-f_i)(x_j-f_j)\rangle_{\mathrm{m}}] \\
&= w_{ij} + \eta[\langle x_ix_j\rangle_{\mathrm{d}} - \langle x_ix_j\rangle_{\mathrm{m}} - f_i\nabla b_j - f_j\nabla b_i] \\
&= w_{ij} + \eta[\mathrm{Cov}_{\mathrm{d}}(x_i,x_j) - \mathrm{Cov}_{\mathrm{m}}(x_i,x_j) + (\langle x_i\rangle_{\mathrm{dm}} - f_i)\nabla b_j \\
&\quad +(\langle x_j\rangle_{\mathrm{dm}} - f_j)\nabla b_i]. \qquad \text{(B.1)}
\end{aligned}$$

We use a shorthand $\nabla_{\mathbf{f}}w_{ij} = \langle x_ix_j\rangle_{\mathrm{d}} - \langle x_ix_j\rangle_{\mathrm{m}} - f_i\nabla b_j - f_j\nabla b_i$ for the resulting gradient when using the transformation $\mathbf{f}$.

The three-step update for the bias is

$$\begin{aligned}
b_i &\leftarrow (-1)^{f_i}\left\{\tilde{b}_i + \eta(\langle\tilde{x}_i\rangle_{\mathrm{d}} - \langle\tilde{x}_i\rangle_{\mathrm{m}}) + \sum_j f_j[\tilde{w}_{ij} + \eta(\langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{d}} - \langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{m}})]\right\} \\
&= (-1)^{f_i}\left\{(-1)^{f_i}\left(b_i + \sum_j f_jW_{ij}\right) + \eta\left[\left\langle x_i^{1-f_i}(1-x_i)^{f_i}\right\rangle_{\mathrm{d}}\right.\right. \\
&\quad \left.\left.-\left\langle x_i^{1-f_i}(1-x_i)^{f_i}\right\rangle_{\mathrm{m}}\right] + \sum_j f_j(-1)^{f_i+f_j}[w_{ij} + \eta\nabla_{\mathbf{f}}w_{ij}]\right\} \\
&= b_i + \sum_j f_jw_{ij} + \eta(\langle x_i\rangle_{\mathrm{d}} - \langle x_i\rangle_{\mathrm{m}}) + \sum_j f_j(-w_{ij} - \eta\nabla_{\mathbf{f}}w_{ij}) \\
&= b_i + \eta\left(\nabla b_i - \sum_j f_j\nabla_{\mathbf{f}}w_{ij}\right). \qquad \text{(B.2)}
\end{aligned}$$

From equations B.1 and B.2, we note that the transformation vectors $\mathbf{f}$ do not cancel out, and thus we end up with different parameters after the update depending on the transformation.

Appendix C: Enhanced Gradient

This appendix presents the derivation of the enhanced gradient update rules 3.6 to 3.8, starting from the transformation-dependent gradient update rules derived in the previous section.

The enhanced gradient is a weighted average of all possible transformation-dependent updates with different transformation vectors **f**. We choose the weights for the different updates inspired by equation B.1 to be

$$\prod_i \langle x_i \rangle_{\mathrm{dm}}^{f_i} (1 - \langle x_i \rangle_{\mathrm{dm}})^{1-f_i},$$

which sum up to one when considering all the exponentially many alternative transformation vectors **f**.

The enhanced gradient for the weights is derived as follows:

$$\begin{aligned}
\nabla_e w_{ij} &= \sum_{\mathbf{f}\in\{0,1\}^n} \left[\prod_k \langle x_k \rangle_{\mathrm{dm}}^{f_k} (1 - \langle x_k \rangle_{\mathrm{dm}})^{1-f_k} \right] [\mathrm{Cov_d}(x_i, x_j) - \mathrm{Cov_m}(x_i, x_j) \\
&\quad + (\langle x_i \rangle_{\mathrm{dm}} - f_i)\nabla b_j + (\langle x_j \rangle_{\mathrm{dm}} - f_j)\nabla b_i] \\
&= \sum_{f_i, f_j \in \{0,1\}} \prod_{k, i\neq k\neq j} [\langle x_k \rangle_{\mathrm{dm}} + (1 - \langle x_k \rangle_{\mathrm{dm}})] \\
&\quad \times \langle x_i \rangle_{\mathrm{dm}}^{f_i} (1 - \langle x_i \rangle_{\mathrm{dm}})^{1-f_i} \langle x_j \rangle_{\mathrm{dm}}^{f_j} (1 - \langle x_j \rangle_{\mathrm{dm}})^{1-f_j} [\mathrm{Cov_d}(x_i, x_j) \\
&\quad - \mathrm{Cov_m}(x_i, x_j) + (\langle x_i \rangle_{\mathrm{dm}} - f_i)\nabla b_j + (\langle x_j \rangle_{\mathrm{dm}} - f_j)\nabla b_i] \\
&= \sum_{f_i, f_j \in \{0,1\}} \langle x_i \rangle_{\mathrm{dm}}^{f_i} (1 - \langle x_i \rangle_{\mathrm{dm}})^{1-f_i} \langle x_j \rangle_{\mathrm{dm}}^{f_j} (1 - \langle x_j \rangle_{\mathrm{dm}})^{1-f_j} \\
&\quad \times [\mathrm{Cov_d}(x_i, x_j) - \mathrm{Cov_m}(x_i, x_j) + (\langle x_i \rangle_{\mathrm{dm}} - f_i)\nabla b_j \\
&\quad + (\langle x_j \rangle_{\mathrm{dm}} - f_j)\nabla b_i] \\
&= \mathrm{Cov_d}(x_i, x_j) - \mathrm{Cov_m}(x_i, x_j) \\
&\quad + (1 - \langle x_i \rangle_{\mathrm{dm}})(1 - \langle x_j \rangle_{\mathrm{dm}})[\langle x_i \rangle_{\mathrm{dm}} \nabla b_j + \langle x_j \rangle_{\mathrm{dm}} \nabla b_i] \\
&\quad + \langle x_i \rangle_{\mathrm{dm}} (1 - \langle x_j \rangle_{\mathrm{dm}})[(\langle x_i \rangle_{\mathrm{dm}} - 1)\nabla b_j + \langle x_j \rangle_{\mathrm{dm}} \nabla b_i] \\
&\quad + (1 - \langle x_i \rangle_{\mathrm{dm}}) \langle x_j \rangle_{\mathrm{dm}} [\langle x_i \rangle_{\mathrm{dm}} \nabla b_j + (\langle x_j \rangle_{\mathrm{dm}} - 1)\nabla b_i] \\
&\quad + \langle x_i \rangle_{\mathrm{dm}} \langle x_j \rangle_{\mathrm{dm}} [(\langle x_i \rangle_{\mathrm{dm}} - 1)\nabla b_j + (\langle x_j \rangle_{\mathrm{dm}} - 1)\nabla b_i] \\
&= \mathrm{Cov_d}(x_i, x_j) - \mathrm{Cov_m}(x_i, x_j).
\end{aligned}$$

For the bias term, it would be possible to derive similarly. However, rather than using a different weight update $\nabla_{\mathbf{f}}\mathbf{W}$ for each bias update, we use the enhanced gradient for the weights when doing each bias update, that is, replacing equation B.2 by $b_i \leftarrow b_i + \eta(\nabla b_i - \sum_j f_j \nabla_e w_{ij})$. Now we get the final formulation of the enhanced gradient, which was presented in equations 3.7 and 3.8, for the bias:

$$\begin{aligned}
\nabla_e b_i &= \sum_{\mathbf{f}\in\{0,1\}^n} \left[\prod_k \langle x_k\rangle_{\mathrm{dm}}^{f_k}(1-\langle x_k\rangle_{\mathrm{dm}})^{1-f_k}\right]\left(\nabla b_i - \sum_j f_j \nabla_e w_{ij}\right) \\
&= \nabla b_i - \sum_j \sum_{f_i,f_j\in\{0,1\}} \langle x_i\rangle_{\mathrm{dm}}^{f_i}(1-\langle x_i\rangle_{\mathrm{dm}})^{1-f_i}\langle x_j\rangle_{\mathrm{dm}}^{f_j}(1-\langle x_j\rangle_{\mathrm{dm}})^{1-f_j} f_j \nabla_e w_{ij} \\
&= \nabla b_i - \sum_j \langle x_j\rangle_{\mathrm{dm}}[(1-\langle x_i\rangle_{\mathrm{dm}})\nabla_e w_{ij} + \langle x_i\rangle_{\mathrm{dm}}\nabla_e w_{ij}] \\
&= \nabla b_i - \sum_j \langle x_j\rangle_{\mathrm{dm}} \nabla_e w_{ij}.
\end{aligned}$$

Appendix D: Invariance of the Enhanced Gradient

Theorem D.1. *The enhanced gradient*

$$\begin{aligned}
\nabla_e w_{ij} &= \mathrm{Cov}_d(x_i, x_j) - \mathrm{Cov}_m(x_i, x_j) \\
\nabla_e b_i &= \nabla b_i - \sum_j \langle x_j\rangle_{dm} \nabla_e w_{ij}
\end{aligned}$$

is invariant to the bit-flipping transformations as described in appendix A.

Proof. We again compose a three-step update consisting of a transformation, update by enhanced gradient, and transformation back. If the resulting model is the same regardless of the transformation vector $\mathbf{f}$, we have proven the claim. The combined update for the weights is

$$\begin{aligned}
w_{ij} &\leftarrow (-1)^{f_i+f_j}[\tilde{w}_{ij} + \eta(\mathrm{Cov}_{\mathrm{d}}(\tilde{x}_i, \tilde{x}_j) - \mathrm{Cov}_{\mathrm{m}}(\tilde{x}_i, \tilde{x}_j))] \\
&= w_{ij} + (-1)^{f_i+f_j}\eta\big[\mathrm{Cov}_{\mathrm{d}}\big(x_i^{1-f_i}(1-x_i)^{f_i}, x_j^{1-f_j}(1-x_j)^{f_j}\big) \\
&\quad - \mathrm{Cov}_{\mathrm{m}}\big(x_i^{1-f_i}(1-x_i)^{f_i}, x_j^{1-f_j}(1-x_j)^{f_j}\big)\big] \\
&= w_{ij} + \eta[\mathrm{Cov}_{\mathrm{d}}(x_i, x_j) - \mathrm{Cov}_{\mathrm{m}}(x_i, x_j)].
\end{aligned}$$

The combined update for the bias is

$$
\begin{aligned}
b_i &\leftarrow (-1)^{f_i}\left[\tilde{b}_i + \eta\nabla_e\tilde{b}_i + \sum_j f_j(\tilde{w}_{ij} + \eta\nabla_e\tilde{w}_{ij})\right] \\
&= (-1)^{f_i}\Bigg\{(-1)^{f_i}\left(b_i + \sum_j f_j w_{ij}\right) + \eta\Bigg[\langle\tilde{x}_i\rangle_{\mathrm{d}} - \langle\tilde{x}_i\rangle_{\mathrm{m}} \\
&\quad - \sum_j \langle\tilde{x}_j\rangle_{\mathrm{dm}}(\langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{d}} - \langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{m}} - \langle\tilde{x}_j\rangle_{\mathrm{dm}}\langle\tilde{x}_i\rangle_{\mathrm{d}} + \langle\tilde{x}_j\rangle_{\mathrm{dm}}\langle\tilde{x}_i\rangle_{\mathrm{m}} \\
&\quad - \langle\tilde{x}_i\rangle_{\mathrm{dm}}\langle\tilde{x}_j\rangle_{\mathrm{d}} + \langle\tilde{x}_i\rangle_{\mathrm{dm}}\langle\tilde{x}_j\rangle_{\mathrm{m}})\Bigg] + \sum_j f_j[(-1)^{f_i+f_j}w_{ij} \\
&\quad + \eta(\langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{d}} - \langle\tilde{x}_i\tilde{x}_j\rangle_{\mathrm{m}} - \langle\tilde{x}_i\rangle_{\mathrm{d}}\langle\tilde{x}_j\rangle_{\mathrm{d}} + \langle\tilde{x}_i\rangle_{\mathrm{m}}\langle\tilde{x}_j\rangle_{\mathrm{m}})]\Bigg\} \\
&= b_i + \eta\Bigg\{\langle x_i - f_i\rangle_{\mathrm{d}} - \langle x_i - f_i\rangle_{\mathrm{m}} - \sum_j[\langle x_j - f_j\rangle_{\mathrm{dm}}(\langle(x_i - f_i)(x_j - f_j)\rangle_{\mathrm{d}} \\
&\quad - \langle(x_i - f_i)(x_j - f_j)\rangle_{\mathrm{m}} - \langle x_j - f_j\rangle_{\mathrm{dm}}\langle x_i - f_i\rangle_{\mathrm{d}} + \langle x_j - f_j\rangle_{\mathrm{dm}}\langle x_i - f_i\rangle_{\mathrm{m}} \\
&\quad - \langle x_i - f_i\rangle_{\mathrm{dm}}\langle x_j - f_j\rangle_{\mathrm{d}} + \langle x_i - f_i\rangle_{\mathrm{dm}}\langle x_j - f_j\rangle_{\mathrm{m}}) \\
&\quad + f_j(\langle x_i x_j\rangle_{\mathrm{d}} - \langle x_i x_j\rangle_{\mathrm{m}} - \langle x_i\rangle_{\mathrm{d}}\langle x_j\rangle_{\mathrm{d}} + \langle x_i\rangle_{\mathrm{m}}\langle x_j\rangle_{\mathrm{m}})]\Bigg\} \\
&= b_i + \eta\Bigg\{\nabla b_i - \sum_j[(\langle x_j\rangle_{\mathrm{dm}} - f_j)(\nabla w_{ij} - f_j\nabla b_i - f_i\nabla b_j - \langle x_i\rangle_{\mathrm{dm}}\nabla b_j \\
&\quad + f_i\nabla b_j - \langle x_j\rangle_{\mathrm{dm}}\nabla b_i + f_j\nabla b_i) + f_j(\nabla w_{ij} - \langle x_i\rangle_{\mathrm{dm}}\nabla b_j - \langle x_j\rangle_{\mathrm{dm}}\nabla b_i)]\Bigg\} \\
&= b_i + \eta\Bigg[\nabla b_i - \sum_j(\langle x_j\rangle_{\mathrm{dm}}\nabla w_{ij} - \langle x_j\rangle_{\mathrm{dm}}\langle x_i\rangle_{\mathrm{dm}}\nabla b_j - \langle x_j\rangle_{\mathrm{dm}}^2\nabla b_i \\
&\quad - f_j\nabla w_{ij} + f_j\langle x_i\rangle_{\mathrm{dm}}\nabla b_j + f_j\langle x_j\rangle_{\mathrm{dm}}\nabla b_i \\
&\quad + f_j\nabla w_{ij} - f_j\langle x_i\rangle_{\mathrm{dm}}\nabla b_j - f_j\langle x_j\rangle_{\mathrm{dm}}\nabla b_i)\Bigg] \\
&= b_i + \eta\Bigg[\nabla b_i - \sum_j\langle x_j\rangle_{\mathrm{dm}}(\nabla W_{ij} - \langle x_i\rangle_{\mathrm{dm}}\nabla b_j - \langle x_j\rangle_{\mathrm{dm}}\nabla b_i)\Bigg] \\
&= b_i + \eta\left(\nabla b_i - \sum_j\langle x_j\rangle_{\mathrm{dm}}\nabla_e W_{ij}\right).
\end{aligned}
$$

References

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*, 1–127.

Bengio, Y., & Delalleau, O. (2009). Justifying and generalizing contrastive divergence. *Neural Comput.*, *21*, 1601–1621.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Carreira-Perpiñán, M. A., & Hinton, G. E. (2005). On contrastive divergence learning. In R. G. Cowell & Z. Ghahramani (Eds.), *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics* (pp. 33–40). Society for Artificial Intelligence and Statistics.

Cho, K., Ilin, A., & Raiko, T. (2011). Improved learning of Gaussian-Bernoulli restricted Boltzmann machines. In *Proceedings of the Twentieth International Conference on Artificial Neural Networks*. Berlin: Springer-Verlag.

Cho, K., Raiko, T., & Ilin, A. (2010). Parallel tempering is efficient for learning restricted Boltzmann machines. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 3246–3253). Piscataway, NJ: IEEE.

Cho, K., Raiko, T., & Ilin, A. (2011). Gaussian-Bernoulli deep Boltzmann machine. In *NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning*, Sierra Nevada, Spain.

Desjardins, G., Courville, A., & Bengio, Y. (2010). Adaptive parallel tempering for stochastic maximum likelihood learning of RBMs. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*.

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., & Delalleau, O. (2009). *Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines* (Tech. Rep. 1345). Montreal: Universitée de Montréal.

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., & Delalleau, O. (2010). Parallel tempering for training of restricted Boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 145–152). Cambridge, MA: MIT Press.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, *11*, 625–660.

Fischer, A., & Igel, C. (2010). Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III* (pp. 208–217). Berlin: Springer-Verlag.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comput.*, *14*, 1771–1800.

Hinton, G. E. (2010). *A practical guide to training restricted Boltzmann machines* (Tech. Rep. 2010-003). Toronto: Department of Computer Science, University of Toronto.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Krizhevsky, A. (2010). *Convolutional deep belief networks on CIFAR-10* (Tech. Rep.). Toronto: Computer Science Department, University of Toronto.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document Recognition. In *Proceedings of the IEEE* (Vol. 86, pp. 2278–2324). Piscataway, NJ: IEEE.

Lee, H., Ekanadham, C., & Ng, A. (2008). Sparse deep belief net model for visual area V2. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems, 20* (pp. 873–880). Cambridge, MA: MIT Press.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 609–616). New York: ACM.

Marlin, B. M., Swersky, K., Chen, B., & de Freitas, N. (2010). Inductive principles for restricted Boltzmann machine learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 509–516). Cambridge, MA: MIT Press.

Raiko, T., Cho, K., & Ilin, A. (2011). *Derivations on the enhanced gradient for the Boltzmann machine* (Tech. Rep.). Espoo: Aalto University School of Science, Department of Information and Computer Science.

Ranzato, M. A., Huang, F. J., Boureau, Y. L., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). Los Alamitos, CA: IEEE.

Salakhutdinov, R. (2009). Learning in Markov random fields using tempered transitions. In Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems, 22* (pp. 1598–1606). Red Hook, NY: Curran.

Salakhutdinov, R., & Hinton, G. E. (2009). Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics* (Vol. 5, pp. 448–455). Cambridge, MA: MIT Press.

Schulz, H., Müller, A., & Behnke, S. (2010). Investigating convergence of restricted Boltzmann machine learning. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*.

Smolensky, P. (1986). Information processing in dynamical systems: foundations of harmony theory. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 1: Foundations* (pp. 194–281). Cambridge, MA: MIT Press.

Tieleman, T. (2008). *Training restricted Boltzmann machines using approximations to the likelihood gradient*. New York: ACM.

Younes, L. (1989). Parametric inference for imperfectly observed Gibbsian fields. *Probability Theory and Related Fields*, *82*, 625–645. doi: 10.1007/BF00341287

Received September 21, 2011; accepted August 13, 2012.