# Enhanced Gradient and Adaptive Learning Rate 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, Finland

## Abstract

Boltzmann machines are often used as building blocks in greedy learning of deep networks. However, training even a simplified model, known as restricted Boltzmann machine (RBM), can be extremely laborious: Traditional learning algorithms often converge only with the right choice of the 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 at plateaus or even diverge. In this work, we present an enhanced gradient which is derived such that it is invariant to bit-flipping transformations. We also propose a way to automatically adjust the learning rate by maximizing a local likelihood estimate. Our experiments confirm that the proposed improvements yield more stable training of RBMs.

## 1. Introduction

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

Deep neural networks are characterized by the large number of layers of neurons and by using layer-wise unsupervised pretraining to learn a probabilistic model for the 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 RBM. Layer-wise pretraining of RBMs then facilitates finding a more accurate model for the data. Various papers (Salakhutdinov & Hinton, 2009; Hinton & Salakhutdinov, 2006; Erhan et al., 2010) empirically confirmed that such multi-stage learning works better than conventional learning methods, such as the backpropagation with random initialization. It is thus important to have an efficient method for training RBM.

Unfortunately, training RBM is known to be difficult. Recent research suggests that without careful choice of learning parameters that are well suited to specific data sets and RBM structures, traditional learning algorithms may fail to model the data distribution correctly (Schulz et al., 2010; Fischer & Igel, 2010; Desjardins et al., 2010). This problem is often manifested in the fact that likelihood decreases during learning.

In this paper, we discuss the difficulties of training RBMs using the traditional gradient and propose a new training algorithm. The proposed improvements include an adaptive learning rate and a new enhanced gradient estimate. The adaptation rule for the learning rate is derived from maximizing a local approximation of the likelihood. The enhanced gradient is designed such that it does not contain the terms which often distract learning using the traditional gradient. The new gradient is also invariant to the data representation.

We conduct extensive experiments comparing the conventional learning algorithms with the proposed one. We use the MNIST handwritten digits data set (LeCun et al., 1998) and the Caltech 101 Silhouettes data set (Marlin et al., 2010) as benchmark problems. Some experiments were performed on the transformed MNIST data set in which each bit was flipped. We refer to this data set as 1-MNIST.

Appearing in *Proceedings of the $28^{th}$ International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

The data set 1-MNIST is known to be more difficult to learn, and we give an explanation fot this effect. The empirical results suggest that the new learning rules can avoid many difficulties in training RBMs.

## 2. Training Restricted Boltzmann Machines

RBM is a stochastic recurrent neural network consisting of binary neurons arranged in two layers (Smolensky, 1986). Each neuron $v_i$ in the visible layer is connected to all the hidden neurons, and each neuron $h_j$ in the hidden layer is connected to all the visible neurons. We denote by $\mathbf{v}$ a binary column vector containing the states $v_i$ of the visible neurons and similarly by $\mathbf{h}$ a vector of hidden states $h_j$.

The probability of a particular state $(\mathbf{v}, \mathbf{h})$ of the network is defined by the energy which is postulated to be

$$E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = -\mathbf{v}^T\mathbf{W}\mathbf{h} - \mathbf{b}^T\mathbf{v} - \mathbf{c}^T\mathbf{h} \tag{1}$$

where parameters $\boldsymbol{\theta}$ include weights $\mathbf{W} = [w_{ij}]$ and biases $\mathbf{b} = [b_i]$ and $\mathbf{c} = [c_j]$. Parameter $w_{ij}$ is the weight of the synaptic connections between neurons $v_i$ and $h_j$. The probability of a state $(\mathbf{v}, \mathbf{h})$ is

$$P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left[-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right],$$

where $Z(\boldsymbol{\theta})$ is the normalizing constant.

### 2.1. Training

Maximum likelihood estimation of the parameters of RBM can be done using gradient-ascent update with learning rate $\eta$ and the following gradients:

$$\nabla w_{ij} = \langle v_i h_j \rangle_{\mathrm{d}} - \langle v_i h_j \rangle_{\mathrm{m}} \tag{2}$$

$$\nabla b_i = \langle v_i \rangle_{\mathrm{d}} - \langle v_i \rangle_{\mathrm{m}} \tag{3}$$

$$\nabla c_j = \langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}}. \tag{4}$$

We denote by $\langle\cdot\rangle_{\mathrm{d}}$ the expectation over the data, or in other words the distribution $P(\mathbf{h} \mid \{\mathbf{v}^{(t)}\}, \boldsymbol{\theta})$. Similarly, $\langle\cdot\rangle_{\mathrm{m}}$ denotes the expectation over the model distribution $P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})$. We also use a shorthand notation $\langle\cdot\rangle_P$ which is the expectation over the probability distribution $P$.

A practical way to avoid computing the gradients exactly, which is not computationally feasible, is to use Markov-Chain Monte-Carlo (MCMC) sampling methods to compute the expectations $\langle\cdot\rangle_{\mathrm{m}}$ approximately. The restricted structure of RBM makes Gibbs sampling efficient in drawing samples from $P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})$: Given one layer, either visible or hidden, the neurons in the other layer are mutually independent. This makes it possible to sample from the whole layer at once.

Training is typically done using only a subset of data examples for computing the expectations $\langle\cdot\rangle_{\mathrm{d}}$ on each iteration. This subset of training data is usually called mini-batch.

#### 2.1.1. Contrastive Divergence

Contrastive divergence (CD) learning (Hinton, 2002) approximates the true gradient in (2)–(4) by computing $\langle\cdot\rangle_{\mathrm{m}}$ using samples obtained after running $n$ steps of Gibbs sampling starting from each data sample of the corresponding mini-batch. The CD gradient for the weights is approximated as

$$\nabla w_{ij} \approx \langle v_i h_j \rangle_{\mathrm{d}} - \langle v_i h_j \rangle_{P_n} \tag{5}$$

where $P_n$ denotes the distribution after $n$ steps of Gibbs sampling. Even though CD learning is known to be biased (Carreira-Perpiñán & Hinton, 2005), it has been proven to work well in practice.

#### 2.1.2. Parallel Tempering

Parallel tempering (PT) sampling was recently proposed to replace Gibbs sampling for estimating $\langle\cdot\rangle_{\mathrm{m}}$ (Desjardins et al., 2010; Cho et al., 2010). The basic idea of PT is that multiple chains of Gibbs sampling are run for models with different “temperatures”. Every now and then, the samples are swapped between the chains. Chains with higher temperatures correspond to more diffuse distributions and therefore they can produce a greater variety of samples. This facilitates better exploration of the state space.

In this paper, we use PT with a set of $N$ inverse temperatures $0 < \beta_2 < \cdots < \beta_{N-1} < 1$. Temperature $\beta_N = 1$ corresponds to the current RBM model with parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$. Smaller values $\beta_i$ correspond to less restricted models with parameters $\boldsymbol{\theta}_i = (\beta_i\mathbf{W}, \beta_i\mathbf{b}, \beta_i\mathbf{c})$. Thus $\beta_1 = 0$ corresponds to the most diffuse distribution.

We run separate $N$ chains for each sample in the mini-batch. After every step in the chains, swaps are proposed and accepted according to the Metropolis rule. The expectations $\langle\cdot\rangle_{\mathrm{m}}$ are computed from the samples of the chain with $\beta = 1$.

### 2.2. Annealed Importance Sampling

It is desirable to know the actual value of the likelihood which is optimized during learning. If the normalizing constant is known, computing the likelihood is straightforward.

*Annealed importance sampling* (AIS) provides a way to estimate the normalizing constant of RBM (Salakhutdinov, 2009a). AIS is based on *simple importance sampling* (SIS), which uses the fact that the ratio of two normalizing constants for two probability densities $P_A(\mathbf{v}) = P^*_A(\mathbf{v})/Z_A$ and $P_B(\mathbf{v}) = P^*_B(\mathbf{v})/Z_B$ can be computed as:

$$\frac{Z_B}{Z_A} = \left\langle \frac{P^*_B(\mathbf{v})}{P^*_A(\mathbf{v})} \right\rangle_{P_A}. \tag{6}$$

In SIS, (6) is estimated using samples from $P_A(\mathbf{v})$.

Using this idea, AIS estimates the normalizing constant of the model distribution by computing the ratio of the normalizing constants of consecutive intermediate distributions ranging from so-called base distribution and the target distribution.

### 2.3. Difficulties in Training RBMs

The fact that the objective function is very costly to estimate makes training RBM difficult. It is difficult to determine how well the learning is progressing. Furthermore, it is not possible to use advanced optimization methods such as conjugate gradient or even line-search.

Learning is performed using stochastic gradient, and it converges to a local solution. It is generally not feasible to compare different local optima analytically. Schulz et al. (2010) and Fischer & Igel (2010) recently showed that depending on initialization and learning parameters the resulting RBMs can highly vary even for a small data set.

Furthermore, most learning algorithms discussed in the previous section can diverge if the learning parameters are not chosen appropriately (Desjardins et al., 2010; Schulz et al., 2010; Fischer & Igel, 2010). The use of advanced MCMC sampling methods such as PT has been shown to avoid divergence but the likelihood can highly fluctuate in the long run without using the appropriate learning rate scheduling (Desjardins et al., 2010; 2009).

One way to analyze the quality of a trained model is to look at the features (the weights $w_{ij}$) and the bias terms $c_j$ corresponding to different hidden neurons $h_j$. Neurons that have a large bias $c_j$ are most of the time active. They are not very useful because the weights associated to them can be incorporated into the bias term $\mathbf{b}$. Other neurons (e.g. with large negative biases $c_j$) can be always inactive or there can be neurons (with weights $w_{ij}$ close to zero) whose activations are independent of data. Such hidden neurons are also useless because they do not contribute to the modeling capacity of RBM.

Ideally, each hidden neuron should represent a distinct "meaningful" feature, for example, a typical part of an image. We have noticed, however, that very often the hidden neurons tend to learn features that resemble the visible bias term $\mathbf{b}$. This effect is more prominent at the initial stage of learning and for data set in which visible bits are mostly active, such as 1-MNIST.

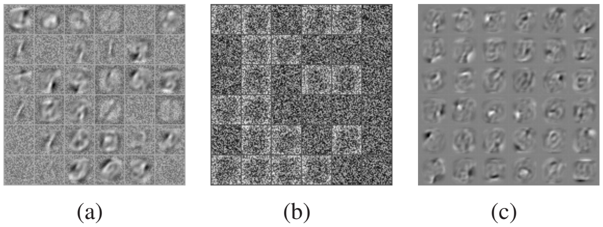

*Figure 1.* Visualization of filters learned after five epochs by RBM with 36 hidden neurons. (a) Traditional gradient, MNIST. (b) Traditional gradient, 1-MNIST. (c) Proposed algorithm (Section 3), 1-MNIST

Fig. 1(a)-(b) show the weights $\mathbf{W}$ of RBM with 36 hidden neurons trained using the traditional gradient (2)–(4) on MNIST and 1-MNIST with the constant learning rate $0.1$ and weights initialized randomly from $\begin{bmatrix}-1 & 1\end{bmatrix}$. The features learned from MNIST look quite good, even though there are some useless neurons. However, the features learned from 1-MNIST are clearly bad: 18 hidden neurons are mostly active and represent global features that somewhat resemble the visible bias, the other 18 neurons are mostly inactive and hence useless.

There is a number of well-known heuristics proposed to improve the training results. They include proper scheduling of the learning rate, weight decay prior for the weights, adding momentum terms to the gradients, and forcing sparsity of the hidden activations. These heuristics are known to help in many practical applications, however, with extra parameters which should be selected very carefully. Good values of these parameters are typically found by trial and error and it seems that one requires a lot of experience to set the learning settings right (Hinton, 2010).

## 3. Improved Training Algorithm

This section describes the two novel contributions.

### 3.1. Adaptive Learning Rate

Here we propose an algorithm for automatically adapting the learning rate while training RBM using stochastic gradient. The automatic adaptation of the learning rate is based on maximizing the local estimate of the likelihood.

Let $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$ be the current model, $\boldsymbol{\theta}' = (\mathbf{W}', \mathbf{b}', \mathbf{c}')$ is the updated model with some learning rate $\eta$ and $P_{\boldsymbol{\theta}}(\mathbf{v}) = P^*_{\boldsymbol{\theta}}(\mathbf{v})/Z_{\boldsymbol{\theta}}$ is the pdf with normalizing constant $Z_{\boldsymbol{\theta}}$ for the model with parameters $\boldsymbol{\theta}$. Now if we assume that the learning rate is small enough and therefore the two models are close to each other, the likelihood of $\boldsymbol{\theta}'$ can be computed as in SIS using (6):

$$P_{\boldsymbol{\theta}'}(\mathbf{v}_{\mathrm{d}}) = \frac{P^*_{\boldsymbol{\theta}'}(\mathbf{v}_{\mathrm{d}})}{Z_{\boldsymbol{\theta}}} \frac{Z_{\boldsymbol{\theta}}}{Z_{\boldsymbol{\theta}'}} = \frac{P^*_{\boldsymbol{\theta}'}(\mathbf{v}_{\mathrm{d}})}{Z_{\boldsymbol{\theta}}} \left\langle \frac{P^*_{\boldsymbol{\theta}'}(\mathbf{v})}{P^*_{\boldsymbol{\theta}}(\mathbf{v})} \right\rangle^{-1}_{P_{\boldsymbol{\theta}}}, \quad (7)$$

where $\mathbf{v}_{\mathrm{d}}$ denotes the training data. In practice, we use samples from the next mini-batch for $\mathbf{v}_{\mathrm{d}}$. [1].

[1] Our experiments showed that if the same samples were used both for obtaining the gradients and the adaptive learning rate, learning rate fluctuated too much in case of PT learning and diverged in case of CD learning .

Now we would like to select a learning rate so as to maximize the likelihood of the new parameters $\boldsymbol{\theta}'$. Equation (7) can be used to approximate the required likelihood. The unnormalized pdf $P^*_{\boldsymbol{\theta}'}$ is computed using the training samples and (1), and the expectation $\langle\cdot\rangle_{P_{\boldsymbol{\theta}}}$ can be estimated using the samples from $P_{\boldsymbol{\theta}}$, like in SIS. These samples are collected in order to estimate the negative term in (5) and therefore computing this expectation can be done practically for free.

In principle, one could find the optimal learning rate that maximizes the local estimate of the likelihood on each iteration. However, this would likely lead to large fluctuations of the learning rate because of the small sample size (minibatch). In our experiments, we selected the new learning rate from the set $\{(1-\epsilon)^2\eta_0, (1-\epsilon)\eta_0, \eta_0, (1+\epsilon)\eta_0, (1+\epsilon)^2\eta_0\}$, where $\eta_0$ is the previous learning rate and $\epsilon$ is a small constant.

### 3.2. Enhanced Gradient

In this section, we propose a new gradient to be used instead of (2)–(4). Let us first define the covariance between two variables under distribution $P$

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

We can rewrite the standard gradient (2) as

$$\begin{aligned}\nabla w_{ij} = {} & \mathrm{cov}_{\mathrm{d}}\left(v_i, h_j\right) - \mathrm{cov}_{\mathrm{m}}\left(v_i, h_j\right) \\ & + \left\langle v_i\right\rangle_{\mathrm{dm}} \nabla c_j + \left\langle h_j\right\rangle_{\mathrm{dm}} \nabla b_i \,, \end{aligned} \tag{8}$$

where $\langle\cdot\rangle_{\mathrm{dm}} = \frac{1}{2}\langle\cdot\rangle_{\mathrm{d}} + \frac{1}{2}\langle\cdot\rangle_{\mathrm{m}}$ is the average activity of neuron under the data and model distributions.

The standard gradient (8) has several potential problems. The gradients w.r.t. the weights contain the terms that point to the same direction as the gradient w.r.t. the bias terms (and vice versa). This effect is prominent when there are many neurons which are mainly active, that is for which $\langle\cdot\rangle_{\mathrm{dm}} \approx 1$. These terms can distract learning of meaningful weights, which often leads to the case when many neurons try to learn features resembling the bias terms, as shown in Fig. 1(b).

When $\langle\cdot\rangle_{\mathrm{dm}} \approx 0$ 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 et al., 2008), which ensure that the average activation of a hidden neuron is kept at low level, have been successful.

A related problem is that the update using (8) is different depending on the data representation. This can be shown by using transformations where some of the binary units of RBM are flipped such that zeros become ones and vice versa:

$$\begin{aligned}\tilde{v}_i &= v_i^{1-f_i^{(v)}}\left(1-v_i\right)^{f_i^{(v)}}, \qquad & f_i^{(v)} \in \{0,1\}\,, \\ \tilde{h}_j &= h_j^{1-f_j^{(h)}}\left(1-h_j\right)^{f_j^{(h)}}, \qquad & f_j^{(h)} \in \{0,1\}.\end{aligned}$$

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

$$\begin{aligned}\tilde{w}_{ij} &= (-1)^{f_i^{(v)}+f_j^{(h)}} w_{ij} \\ \tilde{b}_i &= (-1)^{f_i^{(v)}}\left(b_i + \sum_j f_j^{(h)} w_{ij}\right) \\ \tilde{c}_j &= (-1)^{f_j^{(h)}}\left(c_j + \sum_i f_i^{(v)} w_{ij}\right),\end{aligned}$$

such that the resulting RBM has an equivalent energy function, that is $E(\tilde{\mathbf{x}} \mid \tilde{\boldsymbol{\theta}}) = E(\mathbf{x} \mid \boldsymbol{\theta}) + \mathrm{const}$ for all $\mathbf{x}$. When a model is transformed, updated, and transformed back, the resulting model depends on the transformations:

$$\begin{aligned} w_{ij} &\leftarrow w_{ij} + \eta\Big[\mathrm{cov}_{\mathrm{d}}\left(v_i,h_j\right) - \mathrm{cov}_{\mathrm{m}}\left(v_i,h_j\right) \\ &\qquad + \left(\left\langle v_i\right\rangle_{\mathrm{dm}} - f_i^{(v)}\right)\nabla c_j + \left(\left\langle h_j\right\rangle_{\mathrm{dm}} - f_j^{(h)}\right)\nabla b_i\Big] \end{aligned} \tag{9}$$

$$\begin{aligned} b_i &\leftarrow b_i + \eta\Big[\nabla b_i - \sum_j f_j^{(h)}\left(\nabla w_{ij} - f_i^{(v)}\nabla c_j - f_j^{(h)}\nabla b_i\right)\Big] \\ c_j &\leftarrow c_j + \eta\Big[\nabla c_j - \sum_i f_i^{(v)}\left(\nabla w_{ij} - f_i^{(v)}\nabla c_j - f_j^{(h)}\nabla b_i\right)\Big], \end{aligned}$$

where $\nabla\boldsymbol{\theta}$ are the gradients defined in Eqs. (2)–(4).

We have thus $2^{n_v+n_h}$ different update rules defined by different combinations of binary $f_i^{(v)}$ and $f_j^{(h)}$, $i = 1,\ldots,n_v$ and $j = 1,\ldots,n_h$, where $n_v$, $n_h$ are the number of visible and hidden neurons, respectively. All the update rules are well-founded maximum likelihood updates to the original model. We propose to use as the new gradient a weighted sum of the $2^{n_v+n_h}$ gradients with the following weights:

$$\prod_i \left\langle v_i\right\rangle_{\mathrm{dm}}^{f_i^{(v)}}\left(1-\left\langle v_i\right\rangle_{\mathrm{dm}}\right)^{1-f_i^{(v)}} \prod_j \left\langle h_j\right\rangle_{\mathrm{dm}}^{f_j^{(h)}}\left(1-\left\langle h_j\right\rangle_{\mathrm{dm}}\right)^{1-f_j^{(h)}} \tag{10}$$

By using these weights we prefer sparse data representations for which $\langle\cdot\rangle_{\mathrm{dm}} \approx 0$ because the corresponding models get larger weights.

The proposed weighted sum yields the enhanced gradient

$$\tilde{\nabla} w_{ij} = \mathrm{cov}_{\mathrm{d}}\left(v_i,h_j\right) - \mathrm{cov}_{\mathrm{m}}\left(v_i,h_j\right) \tag{11}$$

$$\tilde{\nabla} b_i = \left\langle v_i\right\rangle_{\mathrm{d}} - \left\langle v_i\right\rangle_{\mathrm{m}} - \sum_j \left\langle h_j\right\rangle_{\mathrm{dm}} \tilde{\nabla} w_{ij} \tag{12}$$

$$\tilde{\nabla} c_j = \left\langle h_j\right\rangle_{\mathrm{d}} - \left\langle h_j\right\rangle_{\mathrm{m}} - \sum_i \left\langle v_i\right\rangle_{\mathrm{dm}} \tilde{\nabla} w_{ij}, \tag{13}$$

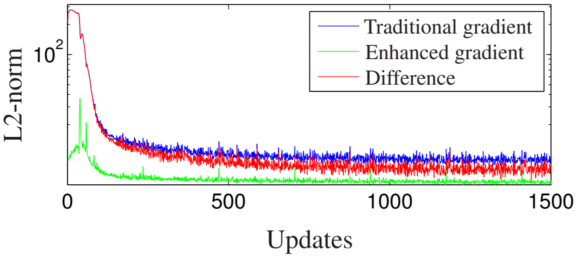

*Figure 2.* L2-norms of the gradients for weights during the learning of a RBM with 361 hidden neurons. The blue curve plots the norms of the traditional gradient, and the green curve plots the norms of the proposed robust gradient. The norms of the difference between two gradients are drawn with the red curve.

in which, by the choice of the weights (10), the effect of the bias gradients in $\nabla w_{ij}$ is canceled out. It can be shown that the new rules are invariant to the bit-flipping transformations. One can also note that the enhanced gradient shares all zeroes with the traditional gradient.

In Figs. 2–3, we present some experimental analysis of the proposed gradient. Fig. 2 shows the norms of the gradient for the weights of an RBM with 361 hidden neurons trained on MNIST data set. It is clear that the additional terms that distract learning dominate in the traditional gradient, especially at the early stage of training.

Fig. 3 shows the differences in the update directions for different neurons of an RBM trained on MNIST. Each element of a matrix is the absolute value of the cosine of the angle between the update directions for the two neurons. The gradients obtained by the traditional rule are highly correlated to each other, especially, at the early stage of learning. On the contrary, the new gradient yields update directions that are close to orthogonal, which allows the neurons to learn distinct features.

## 4. Experiments

In this section, we experimentally compare the proposed improvements to the traditional learning algorithms. In Sections 4.1–4.3, RBMs are trained on the MNIST data set, and in Section 4.4, we use the Caltech 101 Silhouettes data.

We run 20 epochs with a mini-batch size of 128 unless otherwise mentioned. Thus, each RBM was updated about 4,700 times. Both biases **b** and **c** of an RBM were initialized to all zeros. Weights were randomly initialized such that $w_{ij} = \lambda \cdot u$ where $\lambda$ is a weight scale and $u \sim \mathcal{U}(-1, 1)$ denotes a sample from the uniform random variable from $-1$ to 1. By default, we used $\lambda = 1/\sqrt{n_v + n_h}$.

For PT learning, we used 11 different inverse temperatures equally spaced from $\beta_1 = 0$ to $\beta_{11} = 1$. For CD learning, we used $n = 1$ steps of Gibbs sampling. For each setting, RBMs were independently trained with five different initializations of parameters. After training, the normalizing constant of each model was estimated using AIS and the log-probability of the test data was computed. We used $\boldsymbol{\theta}_i = (\beta_i \mathbf{W}, \beta_i \mathbf{b}, \beta_i \mathbf{c})$ and 10,001 equally-spaced temperatures. Each estimate of $Z(\boldsymbol{\theta})$ was averaged over 100 independent AIS runs.

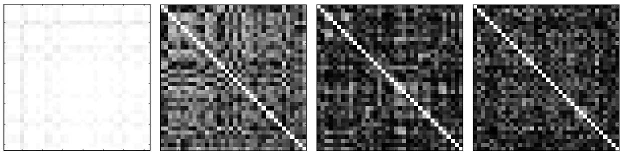

*Figure 3.* The angles between the update directions for the weights of RBM with 36 hidden neurons. White pixels correspond to small angles, while black pixels correspond to orthogonal directions. From left to right: traditional gradient after 26 updates, traditional gradient after 352 updates, enhanced gradient after 26 updates, and enhanced gradient after 352 updates.

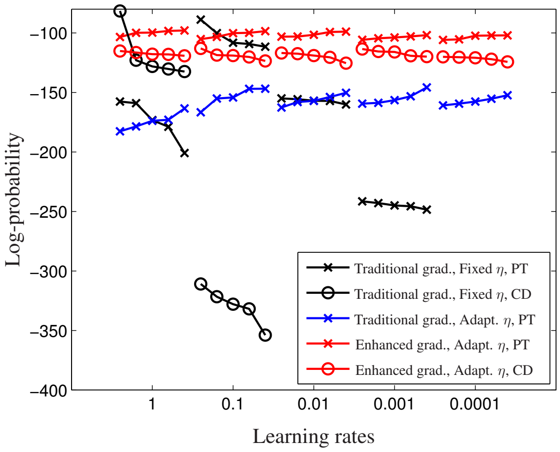

*Figure 4.* Log-probabilities of test data samples computed after 20 epochs for five runs with different initializations for different learning rates. Log-probabilities that do not appear on the plot are smaller than $-400$. The order of the points is arbitrary, and they were sorted in order to make the curves more discriminate.

### 4.1. Sensitivity to Learning Rate

In order to demonstrate how the learning rate can greatly affect training results, we trained RBMs with 361 hidden neurons using the traditional gradient with five different learning rates $\{1, 0.1, 0.01, 0.001, 0.0001\}$. The black curves in Fig. 4 show the log-probability of the test data obtained with PT and CD sampling strategies. It is clear that the resulting RBMs have huge variance depending on the choice of the learning rate. Too small learning rate prevents RBM from learning barely anything, whereas too large learning rate often results in models which are worse than those RBMs trained with proper learning rates. In case of using a learning rate 10, the learning failed completely.

In order to test the proposed adaptive learning rate,

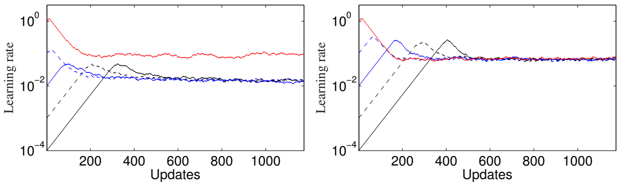

*Figure 5.* Evolution of the adaptive learning rate from five different initializations during learning. The learning rates are shown as a function of the number of updates. RBMs are trained with the traditional gradient (left) and the robust gradient (right).

we trained RBMs with 361 hidden neurons using the traditional gradient and the same five values $\{1, 0.1, 0.01, 0.001, 0.0001\}$ to initialize the learning rate. The blue curves in Fig. 4 show the obtained log-probabilities of the test data. The results are now more stable and the variance among the resulting RBMs is smaller compared to the results obtained with fixed learning rates (the black curves in the same figure). Regardless of the initial learning rate, all RBMs were trained quite well.

These results suggest that the adaptive learning rate works well. However, it was still slightly better to use manually tuned training parameters (using the constant learning rate of $0.1$).

Fig. 5 shows the evolution of the learning rate during learning. Even with very small initial learning rate, the adaptive learning rate could find the appropriate learning after only a few hundred updates. Remarkably, the learning rates converge to the same value when the enhanced gradient is used.

The red curves in Fig. 4 show the log-probabilities of the test data obtained with the new gradient and the adaptive learning rate initialized with five different values. Both PT and CD sampling were tried. It is apparent that the enhanced gradient improves the overall learning performance compared to the traditional gradient. Similar performance was obtained on 1-MNIST (the results are not shown here) because the new gradient is invariant to data representation.

### 4.2. RBM as Feature Extractor

In addition to the log-probabilities of the test data, we trained simple logistic regression classifiers on top of the RBMs to check their feature extracting performance. The activation probabilities of the hidden neurons were used as the features. In order not to destroy the already learned structure of the RBM, no discriminative fine-tuning was performed. This explains why the accuracies reported in this paper are far from the state-of-the-art accuracy on MNIST using deep neural networks (Salakhutdinov, 2009b).

The black curves in Fig. 6 show high variance of the classification results for the traditional gradient depending on the chosen learning rate. The results obtained for MNIST (the upper plot) are pretty good although the choice of the learning does have an effect on performance. However, the classification accuracy obtained for 1-MNIST is very bad, which proves that 1-MNIST is more difficult for training using the traditional gradient.

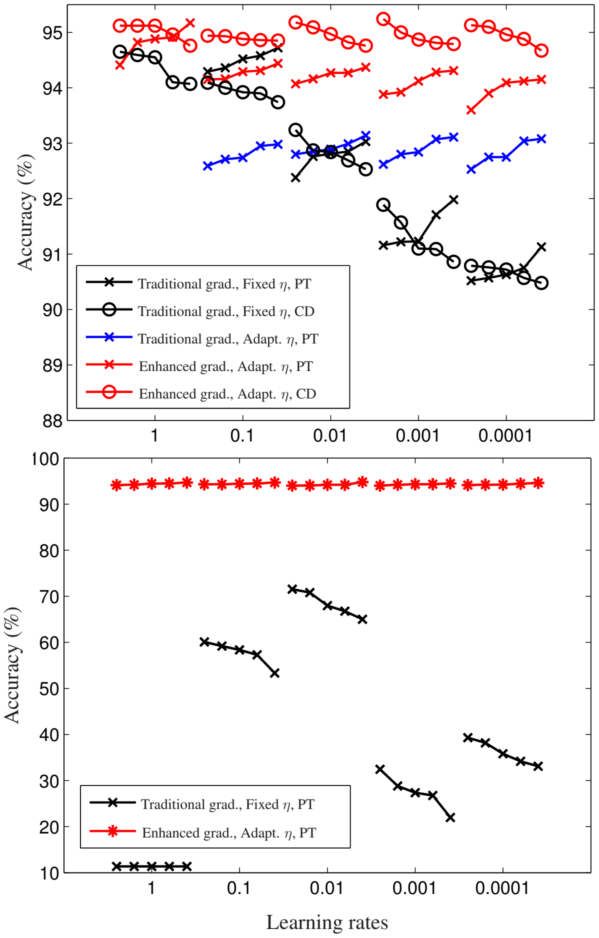

*Figure 6.* Classification accuracy of test data samples computed after 20 epochs for MNIST (above) and 1-MNIST (below). For each initial learning rate, the learning was conducted five times. The results that do not appear on the upper plot were below $88\%$. The order of the points is arbitrary, and they were sorted in order to make the curves more discriminate.

The blue curves in Fig. 6 show that the adaptive learning rate can reduce the variance of the results obtained with the traditional gradient. However, the results were quite significantly worse for the initial learning rate 1.

The red curves in Fig. 6 shows the superior performance of the enhanced gradient and the adaptive learning rate compared to the traditional gradient. Regardless of the initial learning rate, all RBMs leaned features which yielded high classification performance. Note that the results are excellent also for 1-MNIST.

### 4.3. Sensitivity to Weight Initialization

In the next experiment, we test the sensitivity of training results to the scale of the weight initialization. We trained small RBMs with 36 hidden neurons on MNIST using different scales of the initial weights and varying learning rates. PT sampling was used to draw model samples from RBM.

The plot on the left in Fig. 7 visualizes the filters learned by RBMs using the traditional gradient with fixed learning rate. It is clear that the results are highly dependent on the choice of the training parameters: The combination of the initial weight scale and the learning rate should be selected very carefully in order to learn reasonable features. The combination of learning rate $\eta = 0.1$ and weight scale $\lambda = 0.1$ seems to give the best results for the reported experiments. In practice, an optimal combination of the training parameters is usually found by trial and error, which makes training a laborious procedure.

The plot on the right in Fig. 7 shows the filters learned using the new gradient and the adaptive learning rate initialized with five different values. It is clear that the features are much better than the ones obtained with the traditional gradient. Remarkably, no hidden neuron is either dead or always active regardless of the scale of the initial weights and the choice of the initial learning rate.

### 4.4. Caltech 101 Silhouettes

Finally, we tested the proposed learning rules on Caltech 101 Silhouettes data set. RBMs with 500, 1000, and 2000 hidden neurons were trained using the proposed algorithm for 300 epochs with the mini-batch size set to 256. The learning rate was initialized to $0.0001$.

The obtained results are presented in Table 1. Remarkably, the classification accuracy improved by more than 5 % over the best result reported by Marlin et al. (2010).

| Hidden neurons | Log-probability | Accuracy (%) |
|---|---|---|
| 500 | -127.40, -280.91 | 71.56, 68.48 |
| 1000 | -129.69, -190.80 | 72.61, 70.39 |
| 2000 | -131.19, -166.72 | 71.82, 71.39 |

*Table 1.* Log-probabilities and classification accuracies of the test data of Caltech 101 Silhouettes after 300 epochs. First numbers were obtained by PT learning, and the following numbers were by CD learning.

## 5. Discussion

The paper discussed the main difficulties of training RBMs and their underlying reasons. The main reason is that RBMs are best trained using approximate stochastic gradient updates, which leads to high variance in resulting RBMs and possibly diverging behavior. The second reason is that many learning hyperparameters, e.g. learning rate scheduling, have to be carefully and manually chosen depending on the structure of the trained RBMs and the properties of the training data set. Additionally, a problem of having meaningless hidden neurons in RBMs during learning has been demonstrated and discussed.

We proposed a new algorithm for RBM training that addresses the above difficulties. It consists of an adaptive learning rate and an enhanced gradient, and is formulated with well-founded theoretical background. The enhanced gradient could overcome the problem of having hidden neurons learning the near-identical features and was able to speed up the overall learning significantly. Also, unlike the traditional gradient rules which were dependent on the sparsity of the data samples, the enhanced gradient was derived to be invariant to the sparsity could successfully learn very dense data set without any difficulty.

The paper mainly focused on parallel tempering learning, but we also showed that contrastive divergence learning is also improved by adopting the proposed improvements. Our future work will apply the proposed methods to other models in the Boltzmann family, such as deep BMs and Gaussian-Bernoulli BMs.

## Acknowledgments

This work was supported by the honours programme of the department, by the Academy of Finland and by the IST Program of the European Community, under the PASCAL2 Network of Excellence. The authors would like to thank Ilya Sutskever of University of Toronto, Andreas Müller and Hannes Schulz of University of Bonn.

## References

Bengio, Y. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, 2:1–127, January 2009.

Carreira-Perpiñán, M. A. and Hinton, G. E. On Contrastive Divergence Learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados*, pp. 33–40. Society for Artificial Intelligence and Statistics, 2005.

Cho, K., Raiko, T., and Ilin, A. Parallel Tempering is Efficient for Learning Restricted Boltzmann Machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)*, Barcelona, Spain, July 2010.

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., and Delalleau, O. Tempered Markov Chain Monte Carlo for

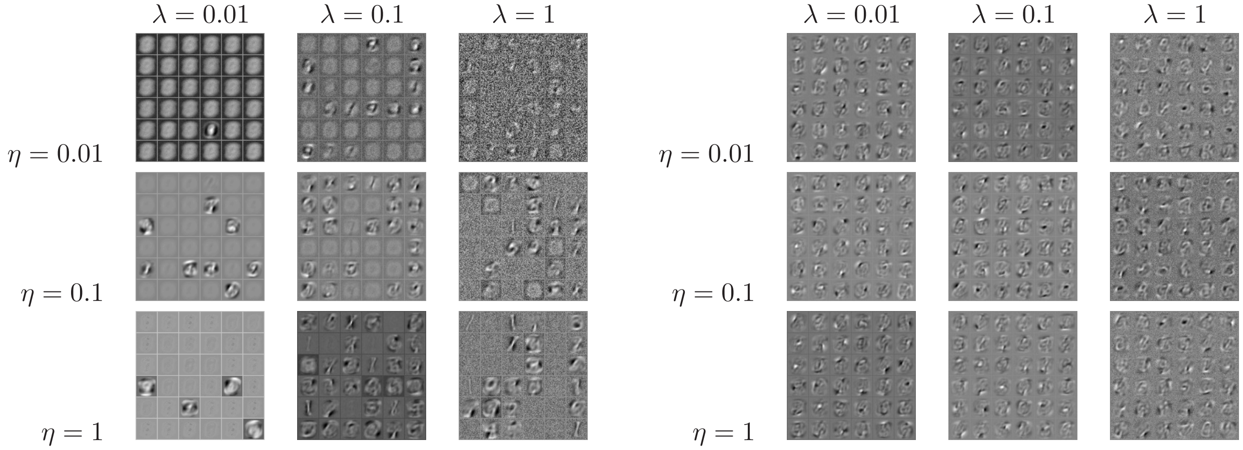

*Figure 7.* Visualization of filters learned by RBMs with 36 hidden neurons on MNIST with various initial learning rates and initial weights scaling. Left: using the traditional gradient with fixed learning rate, right: using the enhanced gradient with adaptive learning rate. Learning was performed for 5 epochs each.

training of Restricted Boltzmann Machines. Technical Report 1345, Universitée de Montréal, 2009.

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., and Delalleau, O. Parallel Tempering for Training of Restricted Boltzmann Machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 145–152, 2010.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11: 625–660, 2010.

Fischer, A. and Igel, C. 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*, ICANN'10, pp. 208–217, Berlin, Heidelberg, 2010. Springer-Verlag.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14:1771–1800, August 2002. ISSN 0899-7667.

Hinton, G. E. A Practical Guide to Training Restricted Boltzmann Machines. Technical Report 2010–003, Department of Computer Science, University of Toronto, 2010.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, volume 86, pp. 2278–2324, 1998.

Lee, H., Ekanadham, C., and Ng, A. Y. Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems 20*, 2008.

Marlin, B. M., Swersky, K., Chen, B., and de Freitas, N. Inductive Principles for Restricted Boltzmann Machine Learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 509–516, 2010.

Salakhutdinov, R. *Learning Deep Generative Models*. PhD thesis, University of Toronto, 2009a.

Salakhutdinov, R. Learning in Markov Random Fields using Tempered Transitions. In *In Advances in Neural Information Processing Systems*, pp. 1598–1606, 2009b.

Salakhutdinov, R. and Hinton, G. E. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pp. 448–455, 2009.

Schulz, H., Müller, A., and Behnke, S. Investigating Convergence of Restricted Boltzmann Machine Learning. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

Smolensky, P. Information processing in dynamical systems: foundations of harmony theory. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pp. 194–281. MIT Press, Cambridge, MA, USA, 1986.