Aalto University
School of Science
Degree Programme of Computer Science and Engineering
Master's Programme in Machine Learning and Data Mining

KyungHyun Cho

Improved Learning Algorithms for Restricted Boltzmann Machines

Master's thesis

Espoo, 14th March 2011

Supervisor: Prof. Juha Karhunen

Instructor: Alexander Ilin, D.Sc. (Tech.) and Tapani Raiko, D.Sc. (Tech.)

Aalto University
School of Science and Technology

Aalto University School of Science Degree Programme of Computer Science and Engineering Master's Programme in Machine Learning and Data Mining	ABSTRACT OF MASTER'S THESIS

Author:	KyungHyun Cho

Title:	Improved Learning Algorithms for Restricted Boltzmann Machines

Number of pages: xii + 84	Date: 14th March 2011	Language: English
Professorship: Computer and Information Science	Code: T-61	

Supervisor:	Prof. Juha Karhunen
Instructor:	Alexander Ilin, D.Sc. (Tech.) and Tapani Raiko, D.Sc. (Tech.)

Abstract:

A restricted Boltzmann machine (RBM) is often used as a building block for constructing deep neural networks and deep generative models which have gained popularity recently as one way to learn complex and large probabilistic models. In these deep models, it is generally known that the layer-wise pretraining of RBMs facilitates finding a more accurate model for the data. It is, hence, important to have an efficient learning method for RBM.

The conventional learning is mostly performed using the stochastic gradients, often, with the approximate method such as contrastive divergence (CD) learning to overcome the computational difficulty. Unfortunately, training RBMs with this approach is known to be difficult, as learning easily diverges after initial convergence. This difficulty has been reported recently by many researchers.

This thesis contributes important improvements that address the difficulty of training RBMs.

Based on an advanced Markov-Chain Monte-Carlo sampling method called parallel tempering (PT), the thesis proposes a PT learning which can replace CD learning. In terms of both the learning performance and the computational overhead, PT learning is shown to be superior to CD learning through various experiments. The thesis also tackles the problem of choosing the right learning parameter by proposing a new algorithm, the adaptive learning rate, which is able to automatically choose the right learning rate during learning.

A closer observation into the update rules suggested that learning by the traditional update rules is easily distracted depending on the representation of data sets. Based on this observation, the thesis proposes a new set of gradient update rules that are more robust to the representation of training data sets and the learning parameters. Extensive experiments on various data sets confirmed that the proposed rules indeed improve learning significantly.

Additionally, a Gaussian-Bernoulli RBM (GBRBM) which is a variant of an RBM that can learn continuous real-valued data sets is reviewed, and the proposed improvements are tested upon it. The experiments showed that the improvements could also be made for GBRBMs.

Keywords: Boltzmann Machine, Restricted Boltzmann Machine, Annealed Importance Sampling, Parallel Tempering, Enhanced Gradient, Adaptive Learning Rate, Gaussian-Bernoulli Restricted Boltzmann Machine, Deep Learning

Acknowledgments

This work has been done in the Department of Information and Computer Science at Aalto University School of Science, as a part of the Master's Programme in Machine Learning and Data Mining (MACADAMIA), and was partly funded by the department through its Summer Internship Program 2010 and Honours programme 2009 and 2010. Prof. Juha Karhunen of the department has supervised this work, and I would like to thank him for his support.

It is impossible for me to express my appreciation to Dr. Alexander Ilin and Dr. Tapani Raiko enough for their enormous help. If it were not them, this thesis would not have been made possible.

For allowing me to work on their CUDA-enabled workstations and valuable discussions, I would like to thank Andreas Müller and Hannes Schulz of University of Bonn, Germany.

I would like to thank my fellow MACADAMIA students and wish best for them all. Also, my three Korean friends–Byungjin Cho, Sungin Cho, and Eunah Cho– in Finland who happen to share the same last name with me have supported me greatly. Thank you.

Thanks to unconditional support from my parents and my little, but taller, brother, I was able to keep my focus entirely on my study.

Lastly, but certainly not least, it would not have been possible for me to survive long, freezing, snowy winter of Finland without songs from four girls of *2NE1* (especially, the leader *CL*).

Espoo, Mar. 14th, 2011

KyungHyun Cho

Contents

List of Abbreviations

AIS	Annealed importance sampling
BM	Boltzmann Machine
CD	Contrastive divergence
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
GBRBM	Gaussian-Bernoulli Restricted Boltzmann Machine
HMC	Hybrid Monte Carlo
ICA	Independent component analysis
MCMC	Markov-Chain Monte-Carlo
MLE	Maximum likelihood estimator
MLP	Multi-layer Perceptron
MPL	Maximum pseudo-likelihood
PCA	Principal component analysis
PCD	Persistent contrastive divergence
pdf	Probability density function
PoE	Product of Expert
PT	Parallel Tempering
RBM	Restricted Boltzmann Machine
RM	Ratio matching
SIS	Simple importance sampling

List of Symbols

$P(\cdot)$	Probability mass/density
$P^*(\cdot)$	Unnormalized probability mass/density
$P_{\boldsymbol{\theta}}(\cdot)$	Probability given the model parameters $\boldsymbol{\theta}$
$P^*_{\boldsymbol{\theta}}(\cdot)$	Unnormalized probability given the model parameters $\boldsymbol{\theta}$
$\langle\cdot\rangle_p$	Expectation over the distribution p
$\mathbf{x}$	A vector of data sample
w_{ij}	A weight connecting i-th and j-th neurons
b_i	A bias term connected to i-th (visible) neuron
c_j	A bias term connected to j-th hidden neuron
$\mathbf{W}$	A weight matrix whose entry at the i-th row and j-th column is w_{ij}
$\mathbf{b}$	A (visible) bias vector
$\mathbf{c}$	A hidden bias vector
$\boldsymbol{\theta}$	A set of parameters for a probabilistic model
$Z(\boldsymbol{\theta})$ or $Z_{\boldsymbol{\theta}}$	A normalizing constant given the model parameter $\boldsymbol{\theta}$
$\mathbf{v}$	A vector of states of visible neurons
$\mathbf{h}$	A vector of states of hidden neurons
$\mathcal{L}(\boldsymbol{\theta})$	A log-likelihood of a model parameter $\boldsymbol{\theta}$
N	The number of training samples
M	The number of samples
P_0 or d	Data distribution
P_∞ or m	Model distribution
P_n	Distribution obtained by running n steps of Gibbs sampling from P_0
η	A learning rate
$\eta_\cdot$	A learning rate for updating $\cdot$
T	(Inverse) temperature ranging from 0 to 1
t	(Inverse) temperature ranging from 0 to 1
K	The number of tempered distributions
$\mathcal{E}(\cdot)$	Reconstruction error

α	Weight decay coefficient
β	Momentum
ρ	Degree of sparsity
n_{swap}	The number of updates between two consecutive swapping
ϵ	Small constant, usually $<< 0$
$\text{Cov}_p(\cdot, \cdot)$	Covariance between two random variables under distribution p
f_k	A flipping transformation of k-th neuron
μ_k	A shifting transformation of k-th neuron
$c(\cdot, \cdot)$	Cosine of the angle between two vectors
λ	Weight scale
$\mathcal{U}(\cdot, \cdot)$	Uniform random variable
$\mathcal{N}(\cdot \mid \mu, \sigma^2)$	Probability of Normal distribution given a mean μ and a variance σ^2
n_v	The number of visible neurons
n_h	The number of hidden neurons
σ_i	Standard deviation of i-th neuron

List of Figures

List of Algorithms

Chapter 1

Introduction

Deep learning has gained its popularity recently as a way of learning complex 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; Ranzato et al., 2010) empirically confirmed that such multi-stage learning works better than conventional learning methods, such as the back-propagation 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, learning algorithms can easily fail to model the data distribution correctly (Schulz et al., 2010; Fischer & Igel, 2010; Desjardins et al., 2010b). This problem is often evidenced by the decreasing likelihood during learning. These failures have discouraged using RBMs and its extensions such as deep Boltzmann machines for more sophisticated and variety of machine learning tasks.

1.1 Contributions of the Thesis

This thesis aims to address this difficulty by proposing advanced learning methods.

Firstly, parallel tempering, an advanced Markov-chain Monte-Carlo sampling algorithm, is proposed to replace a simple Gibbs sampling in obtaining the stochastic gradient. Contrastive divergence learning (Hinton, 2002) which is a learning algorithm for RBM based on Gibbs sampling has been successfully used, in practice, for training RBMs, however, with shortcomings that are discussed later in the thesis. As a way for addressing those shortcomings, parallel tempering learning is proposed and extensively tested through various experiments.

Secondly, the thesis proposes an adaptive learning rate for choosing the appropriate learning rate automatically. The adaptive learning rate is derived from maximizing a local approximation of the likelihood such that it removes the need for manually choosing the learning rate and its scheduling.

Lastly, the enhanced gradient is designed so that the gradients do not contain the terms which often distract learning. Furthermore, the enhanced gradient is invariant to the data representation, for example, a bit-flipping transformation for RBM with both binary visible and hidden neurons, and the sparsity of the data set does not affect learning anymore.

These improvements over the traditional learning methods are extensively studied with various experiments on a number of widely used benchmark data sets. MNIST handwritten digits (LeCun et al., 1998), its bit-flipped version 1-MNIST, OptDigits handwritten digits (Asuncion & Newman, 2007), and Caltech 101 Silhouettes data set (Marlin et al., 2010) are heavily used for testing RBM which is able to model binary data sets. Additionally, a Gaussian-Bernoulli RBM which is a variant of RBM that is capable of modeling continuous values is experimented with CIFAR-10 data set (Krizhevsky, 2009) and CBCL face data set (MIT Center For Biological and Computation Learning).

The experiments along with the theoretical background confirm that the proposed improvements in learning methods indeed remove the discussed difficulties and improve the performance in training RBMs.

1.2 Background and Related Work

A learning algorithm for Boltzmann machine and its variants has been introduced already in 1985 by Ackley et al. (1985). However, training Boltzmann machines was considered

to be difficult due to its stochastic nature and the computational difficulty in estimating the normalizing constant until recently.

The simplest variant of Boltzmann machines, a restricted Boltzmann machine, was introduced by Smolensky (1986). RBM which has no intra-layer connection among the same type of neurons, either visible or hidden, has a big advantage over the fully-connected Boltzmann machine. It became possible to perform Gibbs sampling required for computing the stochastic gradient layer-wise and parallelized.

However, even the parallelized layer-wise Gibbs sampling requires that the sampling needed to be performed until the Gibbs sampling chains converges to the equilibrium. It prevented training RBM on large data sets, because it requires unacceptably long time for generating samples.

In 2002, Hinton (2002) proposed contrastive divergence (CD) learning which can be used for training product-of-expert (PoE) models, one of whose special forms is RBM . CD learning approximates the true gradient by running Gibbs sampling chain for only a few steps starting from the training data samples at each update. This approximate method, however, turned out to work well in practice, and it became the learning method of choice for training RBMs.

Based on the success of CD learning, many variants of it have been introduced since then. Most of them, for instance, persistent contrastive divergence (PCD) learning, can be considered as a variant of stochastic approximation procedure (see e.g. Salakhutdinov, 2009b) which is justified by Younes (1989). The stochastic approximation procedure makes it possible that training RBMs or other types of BMs does not necessarily need to wait for Gibbs sampling chain to converge at every update, thus reducing the computational load.

With these newly proposed learning methods and the introduction of advanced computing techniques, such as GPU Computing (Müller et al., 2010; Bergstra et al., 2010)[1], training RBMs has gained its popularity among researchers and many impressive results have been published (see the rest of the thesis for references). Especially, some papers (see e.g. Hinton & Salakhutdinov, 2006) suggested that a deep neural network, such as a multi-layer perceptron (MLP) with more than three hidden layers, is better trained when each layer is pre-trained separately as if it were a single RBM, which boosted the popularity of deep learning.

Additionally to a simple RBM, many structural variants have been proposed. Semi-restricted Boltzmann machine proposed by Osindero & Hinton (2008) removes a part of restriction

[1]Some of the experiments in the thesis have been performed on GPU Computing using CUV library (Müller et al., 2010): http://www.ais.uni-bonn.de/deep_learning/downloads.html

by introducing the lateral connections among the visible neurons, and it was shown to work well for modeling image patches. A more sophisticated form of RBM which is called Mean-Covariance RBM was introduced by Ranzato et al. (2010) in order to model not only the mean of the visible neurons, but also the covariance among them.

Modifications to the original RBM have been proposed in order to model wider range of data sets. Replacing binary visible units with Gaussian visible units has been proposed earlier and experimented extensively on modeling image patches by Krizhevsky (2009, 2010). Softmax units were successfully introduced as a way for modeling data with a small set of discrete values (Salakhutdinov et al., 2007). Also, recently Nair & Hinton (2010) proposed to use the rectified linear units instead of binary neurons.

Most of the related work presented in this section are separately referenced again throughout the rest of the thesis where the relevant topics are discussed.

1.3 Structure of the Thesis

The main contents of the thesis is split into three chapters. Chapter 2 reviews the concept of Boltzmann machines and restricted Boltzmann machines and how they can be trained using various methods. The chapter, then, discusses several ways to evaluate the trained RBM, such as estimating log-probability of data samples, evaluating the classification accuracy, and visualizing the learned filters. The chapter finishes by stating well known difficulties of training RBMs and conventional remedies that address these issues.

In Chapter 3, the thesis proposes parallel tempering (PT) learning as a substitute for widely used contrastive divergence learning. A basic concept of introducing PT sampling to training RBMs is described, and experimental results supporting the claim that PT learning is superior, are provided.

Chapter 4 proposes two main contributions of this thesis on how to improve learning. They are the enhanced gradient and the adaptive learning rate. Throughout the extensive experiments, the proposed learning methods are shown to address the difficulties presented in Chapter 2.

Chapter 5 describes how RBM can be extended such that it can model continuous valued data sets. A Gaussian-Bernoulli RBM (GB-RBM) is discussed, and several enhancements are proposed. GB-RBMs with the enhancements are tested extensively with various data sets.

Finally, in the last chapter, the overall summary of the thesis is given, and the future work is discussed.

Chapter 2

Restricted Boltzmann Machines

This chapter provides a detailed discussion on Boltzmann machines and its simpler variant, restricted Boltzmann machines. The chapter especially focuses on how to train and assess Boltzmann machines using the stochastic gradient and analyze its difficulties. The conventional remedies and their inherent problems are briefly presented at the end of the chapter.

2.1 Boltzmann Machine

Boltzmann machine (BM) is a stochastic recurrent neural network consisting of binary neurons (Haykin, 1998; Ackley et al., 1985). The network is fully connected, and each connection between two neurons is symmetric such that the effect of one neuron on the state of the other one is symmetric for each pair.

The probability of a particular state $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$ of the network is defined by the energy of BM which is postulated as

$$E(\mathbf{x} \mid \boldsymbol{\theta}) = -\sum_i \sum_{j>i} w_{ij} x_i x_j - \sum_i b_i x_i,$$

where $\boldsymbol{\theta}$ denotes parameters of the network consisting of a weight matrix $\mathbf{W} = [w_{ij}]$ and a bias vector $\mathbf{b} = [b_i]$. w_{ij} is the weight of the synaptic connections between neurons i and j. We assume that $w_{ii} = 0$ and that $w_{ij} = w_{ji}$. The probability of a state $\mathbf{x}$ is, then,

$$P(\mathbf{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left[-E(\mathbf{x} \mid \boldsymbol{\theta})\right] \tag{2.1}$$

where

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \exp\left[-E(\mathbf{x} \mid \boldsymbol{\theta})\right]$$

is the normalizing constant.

It follows from (2.1) that the conditional probability of a single neuron being either 0 or 1 given the states of the other neurons can be written in the following way:

$$P(x_i = 1 \mid \mathbf{x}_{\setminus i}, \mathbf{W}) = \frac{1}{1 + \exp\left(-\sum_{j \neq i} w_{ij} x_j - b_i\right)}, \tag{2.2}$$

where $\mathbf{x}_{\setminus i}$ denotes a vector $[x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_d]^T$.

The neurons of BM are usually divided into visible and hidden ones $\mathbf{x} = [\mathbf{v}^T, \mathbf{h}^T]^T$, where the states $\mathbf{v}$ of the visible neurons are clamped to observed data, and the states $\mathbf{h}$ of the hidden neurons can change freely. In this case of having visible and hidden neurons, the probability of a specific configuration of the visible neurons can be computed by marginalizing out the hidden neurons.

2.1.1 Training Boltzmann Machines

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

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{t=1}^{N} \log P(\mathbf{v}^{(t)} | \boldsymbol{\theta}) = \sum_{t=1}^{N} \log \sum_{\mathbf{h}} P(\mathbf{v}^{(t)}, \mathbf{h} \mid \boldsymbol{\theta}), \tag{2.3}$$

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

The gradient of the log-likelihood is obtained by taking partial derivative of $\mathcal{L}(\boldsymbol{\theta})$ with respect to parameters w_{ij}

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{N}{2}\left[\langle x_i x_j \rangle_{\mathrm{d}} - \langle x_i x_j \rangle_{\mathrm{m}}\right],$$

where a shorthand notation $\langle \cdot \rangle_{P(\cdot)}$ denotes the expectation computed over the probability distribution $P(\cdot)$. Additionally, d and m were used for denoting two probability distributions $P(\mathbf{h} \mid \{\mathbf{v}^{(t)}\}, \boldsymbol{\theta})$ and $P(\mathbf{x} \mid \boldsymbol{\theta})$, respectively. They are the probability of hidden neurons when the visible neurons are clamped to the samples, and the probability of all the

neurons without any fixed neurons. According to the sign of each term, the two terms can be referred to as the positive phase and the negative phase, respectively.

The overall update formula for a parameter w_{ij} is

$$w_{ij} \leftarrow w_{ij} + \eta \left[\langle x_i x_j \rangle_{\mathrm{d}} - \langle x_i x_j \rangle_{\mathrm{m}} \right] , \tag{2.4}$$

where η denotes the learning rate.

For the clarity, from here on we let $\mathbf{b}$ be a vector of biases b_i of the visible neurons only, and $\mathbf{c}$ be a vector of the biases of the hidden neurons. Then, for separate biases of visible and hidden neurons, the update rules are, in analogy to the update rule for the weights,

$$b_i \leftarrow b_i + \eta \left[\langle v_i \rangle_{\mathrm{d}} - \langle v_i \rangle_{\mathrm{m}} \right] , \tag{2.5}$$

and

$$c_j \leftarrow c_j + \eta \left[\langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}} \right] , \tag{2.6}$$

where v_i, h_j, b_i, and c_j denote the i-th visible neuron, the j-th hidden neuron, the i-th visible bias, and the j-th hidden bias.

More details on deriving the update rules are given in Appendix A.

Although the activation and learning rules of BM are both clearly formulated, there are practical limitations in using BM. Especially, the gradient-based update formulas (2.4) – (2.6) are not computationally feasible, as the distributions required in both the positive and negative phases can only be obtained after computing the normalizing constant $Z(\boldsymbol{\theta})$.

Computing $Z(\boldsymbol{\theta})$, however, requires the summation over exponentially many possible configurations of BM, and it is simply impossible for large BMs.

One obvious approach to avoid computing the normalizing constant is to use Markov-Chain Monte-Carlo (MCMC) sampling methods to compute the stochastic gradient. Due to the simplicity of the activation rule for a single neuron given the states of other neurons, a simple Gibbs sampling is enough to get stochastic gradients.

Gibbs sampling can easily be implemented because the conditional distribution of the state of a single neuron in BM given the states of all the other neurons is given by (2.2). A simple description on how to perform Gibbs sampling with BM is described in Algorithm 1.

This approach can greatly reduce the computational burden of the gradient update rules. If it is assumed that the number of samples required for explaining the probability distribution of the whole state space is sufficiently smaller than the size of the state space, that is the

Algorithm 1 Gibbs sampling steps for general BM

Draw $\mathbf{x}_0$ uniformly from the state space.
repeat
  for $i = 1 \dots d$ **do**
    Sample x_i using Equation (2.2).
  end for
until the sufficient number of samples are gathered, or Gibbs sampling has reached the equilibrium.

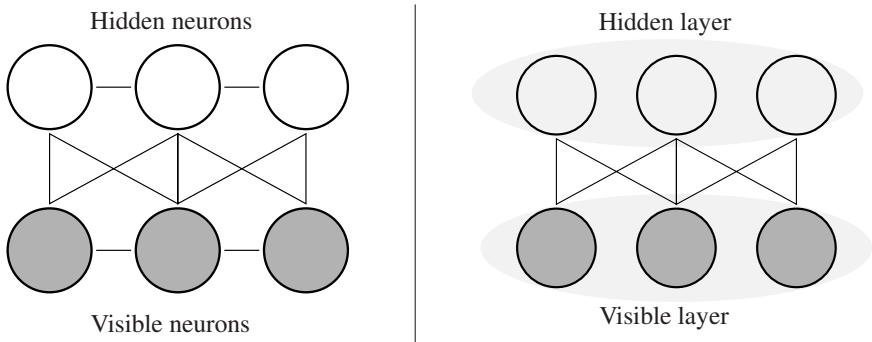
number of all possible combinations of the states of the neurons, the learning of BM is not anymore computational unfeasible.

However, there also exist other kinds of limitations in using Gibbs sampling for training BM. The biggest problem is due to the full-connectivity of BM. Since each neuron is connected to and influenced by all the other neurons, it takes as many steps as the number of neurons to get one sample of the BM state. Even when the visible neurons are clamped to the training data, the number of required steps for a single fresh sample is still at least the number of hidden neurons. This makes the successive samples in the chain highly correlated with each other and this poor mixing affects the performance of learning. Another limitation of this approach is that multi-modal distributions are problematic for Gibbs sampling (Salakhutdinov, 2009b): Due to the nature of component-wise sampling, the samples might miss some modes of the distribution.

2.2 Restricted Boltzmann Machine

To overcome practical limitations imposed on the general Boltzmann machine such as the problem of inefficient sampling, a structurally restricted version of Boltzmann machine called Restricted Boltzmann Machine (RBM) has been proposed by Smolensky (1986). RBM is constructed by removing the lateral connections in-between the visible neurons and the hidden neurons. Therefore, a visible neuron would only have edges connected to the hidden neurons, and a hidden neuron would only have edges connected to the visible neurons. Now, the structure of RBM can be divided into two layers with inter-connecting edges. The relationship between BM and RBM is illustrated in Figure 2.1.

Although the imposed restriction could possibly suggest that the representational power might have been reduced, Le Roux & Bengio (2008) showed that RBM is a universal approximator such that it can model any discrete-valued probability distribution (Le Roux & Bengio, 2008).



(a) Boltzmann Machine (b) Restricted Boltzmann Machine

Figure 2.1: Illustration of the relationship between Boltzmann machine and restricted Boltzmann machine

As the restriction has been imposed on the structure, the energy and the state probability must be modified accordingly:

$$E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = -\mathbf{v}^T \mathbf{W} \mathbf{h} - \mathbf{b}^T \mathbf{v} - \mathbf{c}^T \mathbf{h} \tag{2.7}$$
$$P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right\},$$

where now parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$ include biases $\mathbf{b}$ and $\mathbf{c}$.

Since each hidden neuron is independent of each other given all the visible neurons, it is possible to explicitly sum out the hidden neurons and obtain the unnormalized probability of the visible neurons. The probability of a state of visible neurons $\mathbf{v}$ is, then,

$$P(\mathbf{v} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(\mathbf{b}^T \mathbf{v}) \prod_{j=1}^{n_h} \left(1 + \exp\left(c_j + \sum_{i=1}^{n_v} w_{ij} v_i\right)\right), \tag{2.8}$$

where n_v and n_h are the number of the visible neurons and the hidden neurons, respectively (Salakhutdinov, 2009a).

2.2.1 Training Restricted Boltzmann Machine

The learning rules of RBM , then, become

$$w_{ij} \leftarrow w_{ij} + \eta_w \left[\langle v_i h_j \rangle_{\mathrm{d}} - \langle v_i h_j \rangle_{\mathrm{m}}\right] \tag{2.9}$$
$$b_i \leftarrow b_i + \eta_b \left[\langle v_i \rangle_{\mathrm{d}} - \langle v_i \rangle_{\mathrm{m}}\right] \tag{2.10}$$
$$c_j \leftarrow c_j + \eta_c \left[\langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}}\right], \tag{2.11}$$

where the same shorthand notation $\langle \cdot \rangle_{P(\cdot)}$ was used as before.

Although there is no rigorous theoretical background on choosing learning rates, traditionally, smaller learning rates are used for learning both biases (Hinton, 2010).

Since RBM is a special case of BM, it is possible to employ the same Gibbs sampling to learn. Thanks to its restricted structure, Gibbs sampling can be used more efficiently, as given one layer, either visible or hidden, the neurons in the other layer become mutually independent (see Figure 2.2). This possibility of the layer-wise sampling enables the full utilization of the modern parallelized computing environment.

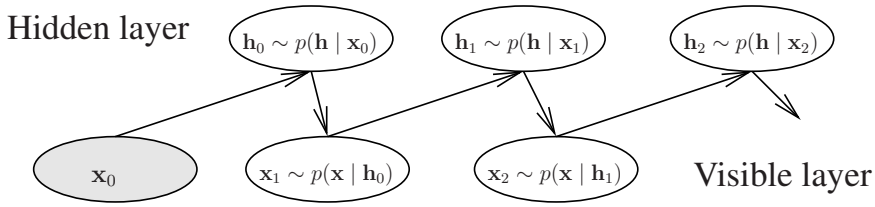


Figure 2.2: Visualization of the idea of how the layer-wise Gibbs sampling is done in RBM .

However, as the number of neurons in RBM increases, a greater number of samples must be gathered by Gibbs sampling in order to properly explain the probability distribution represented by RBM . Moreover, due to the nature of Gibbs samplings, the samples might still miss some modes of the distribution.

Many approaches have been proposed to overcome these difficulties.

2.2.2 Contrastive divergence learning

One popular approach is contrastive divergence (CD) learning proposed by Hinton (2002) as an approximate method for training Product-of-Expert models. Equation (2.8) directly implies that RBM is a special case of PoE models, and CD learning can readily be used for training RBMs.

CD learning approximates the true gradient by replacing the expectation over $P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})$ with an expectation over a distribution P_n that is obtained by running n steps of Gibbs sampling from the empirical distribution defined by the training samples. Figure 2.3 illustrates the distributions P_0 and P_n.

For the weights, the CD learning formula, then, becomes

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

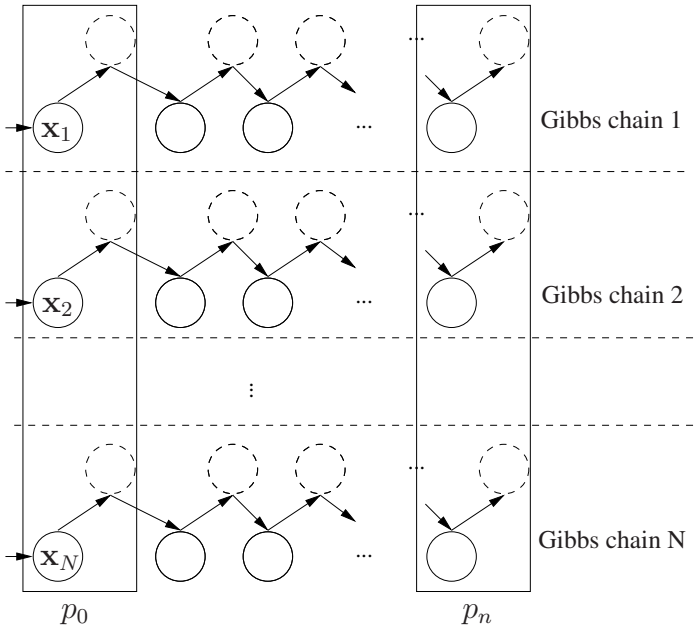


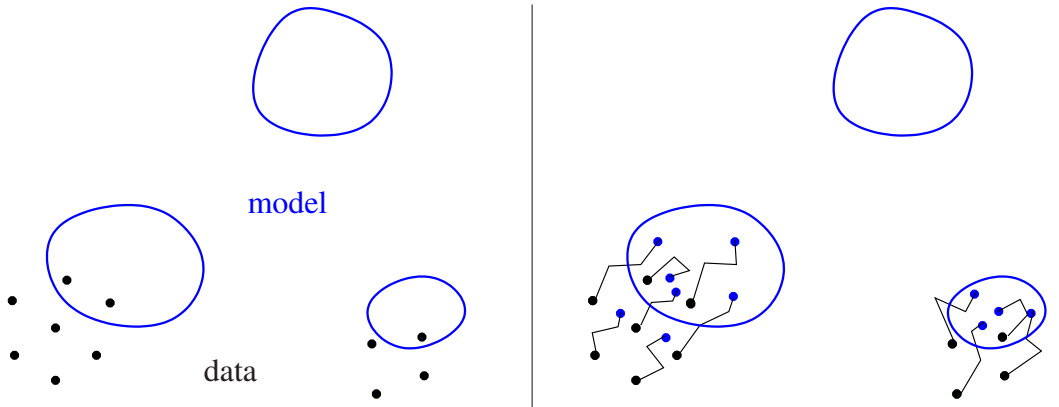
Figure 2.3: Visualization of how CD learning obtains the empirical distribution used in the positive phase and the approximate model distribution used in the negative phase. In the figure, each row represents the Gibbs sampling chain starting from each training data sample, and p_0 and p_n denote the empirical distribution and the approximate model distribution, respectively.

It should be noted that the case $n = 0$ produces the empirical distribution $P(\mathbf{h} \mid \{\mathbf{v}^{(t)}\}, \boldsymbol{\theta})$ used in the positive phase, whereas the case $n = \infty$ produces the true distribution of the negative phase $P(\mathbf{x} \mid \boldsymbol{\theta})$ (Carreira-Perpiñán & Hinton, 2005; Bengio & Delalleau, 2009).

As it can be anticipated from the fact that the direction of the gradient is not identical to the exact gradient, CD learning is known to be biased (Carreira-Perpiñán & Hinton, 2005; Bengio & Delalleau, 2009). Nevertheless, CD learning has been shown to work well in practice. A good property of CD is that in case the data distribution is multi-modal, running the chains starting from each data sample guarantees, that the samples approximating the negative phase have representatives from different modes.

This advantage of CD learning, however, is its disadvantage at the same time. The samples from P_n do not necessarily explain the whole state space. Hence, some of the modes in the model distribution are not explored, and even after learning has converged the model distribution possesses the modes that are not in the data distribution defined by the training data set. This problem is illustrated in Figure 2.4.

In order to overcome this problem, different approaches based on CD learning have been proposed. Among them persistent contrastive divergence (PCD) learning is the simplest extension of CD learning (Tieleman, 2008).



(a) Model distribution and training samples

(b) CD fantasy samples and training samples

Figure 2.4: The left figure shows the model distribution (blue) and and the training samples (black dots). The blue dots in the right figure indicates the fantasy particles obtained by CD learning. It is apparent that the fantasy particles failed to explain the whole space by missing the mode at the top.

At every gradient update step, CD learning performs the Gibbs sampling starting from the training data samples, whereas PCD learning begins the sampling from the model samples obtained at the last gradient update. In this way, it is expected for the model samples to explore the modes in the model distribution that are not close to the training samples.

However, PCD learning still suffers from missing the modes in the model distribution as learning progresses. It is due to the poor mixing of the Gibbs sampling which produces the highly-correlated samples for successive gradient updates. This behavior makes the approaches based on CD learning to suffer from the divergence of the likelihood (Schulz et al., 2010; Fischer & Igel, 2010; Desjardins et al., 2010b,a) if learning is performed without carefully and manually chosen learning heuristics such as learning rate schedule, weight decay, and momentum.

Numerous approaches based on CD learning, other than PCD learning, have been proposed recently. For instance, Fast PCD learning proposed by Tieleman & Hinton (2009) extends PCD learning by maintaining fast weights that help obtaining better model samples.

2.2.3 Learning based on advanced MCMC sampling methods

Instead of approximating the gradient direction, it is possible to apply more sophisticated MCMC sampling methods other than simple Gibbs sampling.

One alternative to the Gibbs sampling is parallel tempering (PT) sampling (Earl & Deem, 2005) which was recently proposed as a replacement for Gibbs sampling in training RBMs

by Desjardins et al. (2010b) and Cho et al. (2010). The detailed description of PT and how PT is used for training RBMs is given in Chapter 3 with the experiments showing the superiority of PT learning compared to CD learning.

In addition to PT learning, other approaches based on advanced MCMC sampling methods have also been proposed. For instance, stochastic approximation procedure based on tempered transition (Neal, 1994) is one that was proposed recently by Salakhutdinov (2009b) that utilizes multiple chains of Gibbs sampling with different temperatures. A hybrid Monte Carlo algorithm (HMC) also has been successful in training more sophisticated RBMs such as a factored 3-way RBM (Ranzato et al., 2010) and an energy-based model (Teh, 2003), recently.

2.2.4 Other approaches

The approaches presented so far are based on the stochastic approximation using MCMC sampling. However, there exist other approaches for training RBMs.

One approach is to approximate the likelihood function with the pseudo-likelihood (Besag, 1975), and thus, training RBMs becomes maximizing the pseudo-likelihood. The log-pseudo-likelihood function given a data set $\{\mathbf{v}^{(t)}\}_{t=1}^{N}$ is defined as

$$f_{\text{PL}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{t=1}^{N} \sum_{i=1}^{d} \log P(\mathbf{v}_i^{(t)} \mid \mathbf{v}_{\setminus i}^{(t)}),$$

where $\mathbf{x}_{\setminus i}$ denotes a vector $[x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_d]^T$ as before. The hidden neurons can be explicitly summed out by Equation (2.8)

The maximum pseudo-likelihood (MPL) learning approximate the joint probability distribution of RBM with the product of one-dimensional probability distributions. Although it removes the necessity of computing the intractable normalizing constant, MPL learning tends not to work well neither with RBMs nor BMs (Marlin et al., 2010; Salakhutdinov, 2009b), as it does not approximate the maximum likelihood estimator (MLE) well except for some extreme cases (Geyer, 1991).

Another approach, ratio matching (RM) was recently proposed by Hyvärinen (2007). Instead of the likelihood, RM considers ratios of probabilities. The data ratio which is defined by the ratio between the probability of a given observation and the probability of the observation vector with one variable i flipped as in Equation (2.12), and the model ratio is the same ratio under the model distribution. RM learning tries to force the data and model ratio

as close as possible.

RM is beneficial as the ratio does not require the computation of the normalizing constant, as

$$\frac{P(\mathbf{v})}{P(\mathbf{v}_{\neg i})} = \frac{P^*(\mathbf{v})}{P^*(\mathbf{v}_{\neg i})}, \tag{2.12}$$

where $\mathbf{v}_{\neg i}$ is equivalent to $\mathbf{v}$ with the i-th component flipped.

Additionally, recently proposed generalized score matching (Lyu, 2009) can be used to train RBMs.

These approaches have been compared to each other and to the stochastic approximation by Marlin et al. (2010). However, these learning methods suffer from the computational complexity when the dimensionality of the observations is large, and they do not show significant improvement over the stochastic approximation based on MCMC sampling. Hence, this thesis only considers stochastic gradient-based method using MCMC sampling.

2.3 Evaluating Restricted Boltzmann Machines

2.3.1 Likelihood and Annealed Importance Sampling

A natural way to assess the performance of a trained RBM is to compute the likelihood of the model and the probabilities of test data samples under the trained RBM. Also, as will be discussed in Chapter 3 and was shown in the author's paper (Cho et al., 2010), the probability of the random data samples also can be used as a measure of the goodness of RBMs.

Due to the structural restriction, explicitly summing out the hidden neurons is fairly straightforward (see Equation (2.8),) however, unfortunately computing the probability of an observation is still intractable due to the normalizing constant. The normalizing constant can only be computed exactly by summing exponentially many terms, and unless the dimensionality of the data set is very small, it is simply impossible. Thus, instead of exactly computing it, an approximate method must be employed.

For estimating the normalizing constant, this thesis uses *annealed importance sampling* (AIS) (Neal, 1998) which has been successfully employed for computing the normalizing constant of RBM (Salakhutdinov, 2009b).

AIS is based on *simple importance sampling* (SIS) method that could estimate the ratio

of two normalizing constants. For two probability densities $P_A(\mathbf{x}) = \frac{P_A^*(\mathbf{x})}{Z_A}$ and $P_B(\mathbf{x}) = \frac{P_B^*(\mathbf{x})}{Z_B}$, the ratio of two normalizing constants Z_A and Z_B can be estimated by a Monte Carlo sampling method without any bias if it is possible to sample from $P_A(\cdot)$:

$$\frac{Z_B}{Z_A} = E_{P_A}\left[\frac{P_B^*(\mathbf{x})}{P_A^*(\mathbf{x})}\right] \approx \frac{1}{M}\sum_{i=1}^{M}\frac{P_B^*(\mathbf{x}_i)}{P_A^*(\mathbf{x}_i)}, \tag{2.13}$$

where $\mathbf{x}_i$ are samples from $P_A(\mathbf{x})$. The quality of the approximation in terms of the variance depends highly on how close $P_A(\cdot)$ and $P_B(\cdot)$ are. If $P_A(\cdot)$ is not near-perfect approximation to P_B, then the variance of the estimate can be as large as infinity.

Based on SIS, 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. The base distribution is chosen such that its normalizing constant Z_0 can be computed exactly and it is possible to collect independent samples from it. A natural choice of the base distribution for RBM is RBM with zero weights $\mathbf{W}$. This yields the normalizing constant

$$Z_0 = \prod_i (1 + \exp\{b_i\}) \prod_j (1 + \exp\{c_j\}),$$

where indices i and j go through all the visible and hidden neurons, respectively.

By computing the product of the estimated ratios of the intermediate normalizing constants and Z_0, the normalizing constant of the target RBM can be estimated. The algorithm implementing AIS is outlined in Algorithm 2.

The presented algorithm describes constructing intermediate RBMs following what Salakhutdinov (2009a) proposed. The base distribution is represented by RBM with zero weights, but biases that are identical to those of the target RBM. However, it should be noticed that there are other possibilities for constructing intermediate distributions and choosing a base distribution. For instance, in the following chapters, the base distribution is an RBM with both zero weights and zero biases such that there is no need for each intermediate RBM to maintain twice as many hidden neurons as the target RBM has.

2.3.2 Classification accuracy and other measures

It is evident from the previously mentioned research papers utilizing deep neural networks built from the stack of RBMs (Salakhutdinov, 2009b; Hinton & Salakhutdinov, 2006) that the hidden activation probabilities of RBM trained on the data set could improve the classi-

Algorithm 2 Estimating the normalizing constant by annealed importance sampling

Create a sequence of temperatures T_k such that $0 = T_0 < T_1 < \cdots < T_K = 1$.
Create a base RBM R_0 with parameters $\boldsymbol{\theta}_0 = (\mathbf{W}_0, \mathbf{b}, \mathbf{c})$, where $\mathbf{W}_0 = 0$.
Create a sequence of intermediate RBMs R_k such that

- It has twice as many hidden nodes as the target RBM has.
- Parameters are $\boldsymbol{\theta}_k = \left(\left[(1 - T_k)\mathbf{W}_0 \;\; T_k\mathbf{W} \right], \left[(1 - T_k)\mathbf{b}_0 \;\; T_k\mathbf{b} \right], \left[(1 - T_k)\mathbf{c}_0^T \;\; T_k\mathbf{c}^T \right]^T \right)$.

for $m = 1 \cdots M$ **do**
  Sample $\mathbf{x}_1$ from R_0.
  for $k = 1 \cdots K - 1$ **do**
    Sample $\mathbf{x}_{k+1}$ from R_k by one-step Gibbs sampling starting from $\mathbf{x}_k$.
  end for
  Set $u_m = \prod_{k=1}^{K} \frac{P_k^*(\mathbf{x}_k)}{P_{k-1}^*(\mathbf{x}_k)}$, where $P_k^*(\cdot)$ is an unnormalized marginal distribution function of R_k.
end for
The estimate of $\frac{Z_K}{Z_0}$ is $\frac{1}{M} \sum_{m=1}^{M} u_m$.

fication accuracy compared to classifying the data set based on its raw features. However, these approaches often require the discriminative fine-tuning which destroys the generative structure of RBM .

Fortunately, recent papers suggest that the hidden activation probabilities of RBM which was trained in a unsupervised manner also help the classification task. Krizhevsky (2009) successfully used a Gaussian-Bernoulli RBM to extract features from images that help obtaining high classification accuracy. Also, more sophisticated forms of RBM introduced recently (Ranzato & Hinton, 2010; Ranzato et al., 2010; Osindero & Hinton, 2008) were shown to be able to extract features that are more useful for the classification task.

Furthermore, Coates et al. (2010) showed that features extracted by the probabilistic models learned in an unsupervised way outperforms the supervised counter-parts such as convolutional neural networks (LeCun et al., 1998) and convolutional deep belief network (Krizhevsky, 2010).

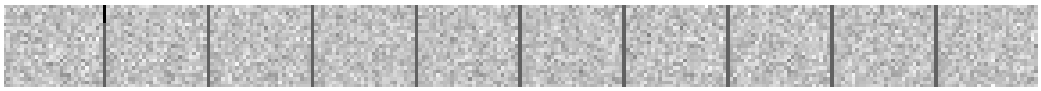
Thus, it is sensible to use the classification accuracy of the trained RBM as a performance measure.

Additionally, thanks to its bipartite structure and the layer-wise Gibbs sampling, the reconstruction error could also be used as a measure for the performance assessment (Hinton, 2010). A reconstruction error is defined as

$$\mathcal{E}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_1\|_2,$$

where $\mathbf{x}_1$ is a sample from $p(\mathbf{x} \mid \mathbf{h}_0, \boldsymbol{\theta})$, and $\mathbf{h}_0$ is a sample from $p(\mathbf{h} \mid \mathbf{x}, \boldsymbol{\theta})$. A simple

(a) Test samples



(b) Before training ($\mathcal{E} = 203.7805$)

(c) After 1 epoch ($\mathcal{E} = 26.5676$)

(d) After 5 epochs ($\mathcal{E} = 20.9808$)

Figure 2.5: Examples of reconstruction errors for RBM with 100 hidden neurons applied to MNIST handwritten digits. The figure shows randomly selected sample digits (a) and their reconstructions (b-d). The reconstruction errors $\mathcal{E}$ on the whole test data set are shown inside the brackets. Reconstructions are shown using the activation probabilities rather than the actual activations which are the samples collected based on the activation probabilities.

example of how reconstruction error decreases during training is given in Figure 2.5.

However, these measures are not directly reflecting the true quality of RBM , since training neither maximizes nor minimizes any of these measures. Therefore, for the rest of this thesis, the experiments mostly assess the trained RBM by the likelihood of the model and the probabilities of the test samples given the model.

2.3.3 Directly visualizing and inspecting parameters

Lastly, 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 of the trained RBM. It especially helps when training data samples consist of images that can be readily visualized.

For instance, features of RBM trained on handwritten digits can be visualized as shown in Figure 2.6. Each feature, or filter, resembles a part of digits, or a combination of parts of digits. When learning fails, it is easy to observe degenerate features that are noisy global features.

In case of hidden biases, the values itself suggest whether each hidden neuron contributes

to the modeling capacity of RBM . Neurons that have a large bias c_j are most of the time active, and they are not very useful, as the weights associated to them can be incorporated into the bias term $\mathbf{b}$. On the other hand, hidden neurons that are mostly inactive (e.g., with large negative biases c_j) or whose activations are independent of data are also useless, as the learning capacity of the RBM does not change even if they are removed.

Like other indirect measures presented previously, the visualization and inspection of parameter values must be performed carefully. There is no objective measure for the quality of the visualized features, and the visualized features and the values of biases may evolve slowly over training.

2.4 Difficulties and conventional remedies

2.4.1 High variance in resulting RBMs and divergence

The fact that the target function cannot be computed exactly during learning makes training RBMs difficult. It is computational infeasible to tell when the learning has converged, or even it is not easy to tell whether the learning is actually happening. Furthermore, it is not possible to use any advanced gradient method such as non-linear conjugate gradient.

Since learning is performed using stochastic gradient updates, it converges to a local solution. The problem is that it is not feasible to compare the different solutions analytically, and choose the best one among them. Schulz et al. (2010) and Fischer & Igel (2010) recently showed that depending on the initialization and the learning parameters the resulting RBMs vary highly even on the small toy data sets.

More problematically, most approximate approaches presented in the previous sections have been shown to diverge, if the learning parameters were not chosen appropriately (Desjardins et al., 2010b; Schulz et al., 2010; Fischer & Igel, 2010). The use of a better MCMC sampling method, e.g. parallel tempering, has been shown to better avoid the diverging behavior, but in a long run without using the appropriate learning rate scheduling, the log-likelihood fluctuates highly (Desjardins et al., 2010b, 2009) which is not desirable.

2.4.2 Existence of possibly meaningless hidden neurons

It has been shown that RBM is a universal approximator so that with enough number of hidden neurons it can model any discrete-valued probability distribution (Le Roux & Bengio,

2008).

However, in practice, the number of hidden neurons is always limited, and depending on learning procedures, not all hidden neurons contribute to the representational power of RBM .

For instance, those hidden neurons that are always active are meaningless, since the weights associated to them can be incorporated into bias terms. Also, any hidden neuron that is inactive always is meaningless, since then, the removal of the hidden neuron does not affect the modeling capacity of RBM at all (see Section 2.3.3 for details on determining meaningless hidden neurons.)

Ideally, each hidden neuron should represent a distinct "meaningful" feature, for example, a typical part of the 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 where each bit of MNIST handwritten data set was flipped.

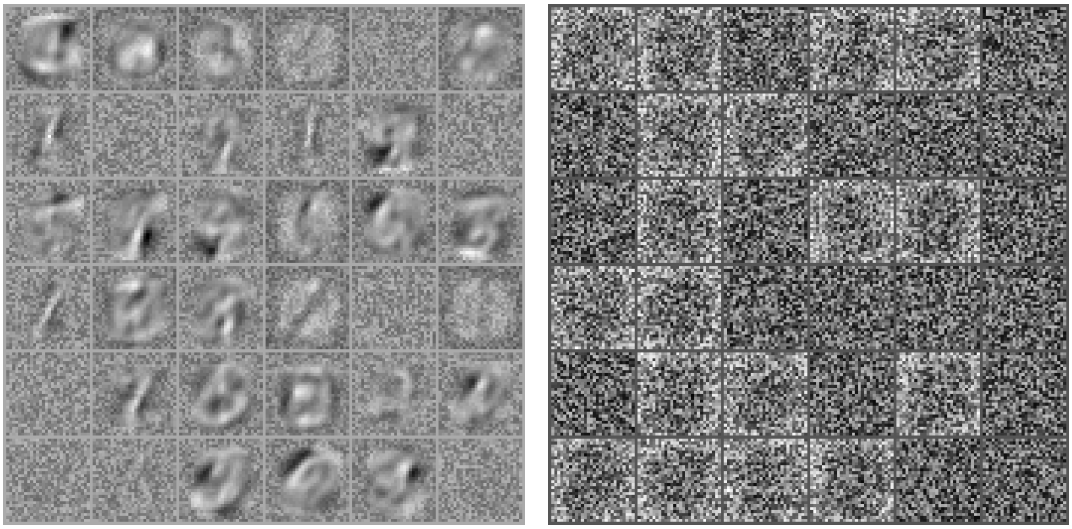
Figure 2.6(b) presents an example how RBM can be ill-trained when the learning parameters were not carefully chosen and the training samples were dense in a sense that the number of ones in each training sample is much more than that of zeros. The RBM with 36 hidden neurons were learned on 1-MNIST which is a very dense data set compared to the original MNIST.

18 hidden neurons were not able to learn any useful features, and they are mostly inactive. The other 18 neurons are mostly active, and as anticipated, learned global features that somewhat resemble the visible bias.

Even when the training data samples are not dense, with the small number of hidden neurons, inappropriate choice of learning parameters, and inappropriate choice of initialization of the parameters, many hidden neurons will be useless. The visualization of the filters learned by RBM with 36 hidden neurons trained on MNIST with the constant learning rate 0.1 and the initial weights sampled from the uniform distribution between -1 and 1 is shown in Figure 2.6(a). In the figure, about 20 neurons out of 36 neurons look as if they learned some useful features. However, there still exist those neurons that are either mostly active or mostly inactive.

2.4.3 Conventional remedies

There is a number of well-known heuristics that are known to yield better training results:



(a) MNIST (b) 1-MNIST

Figure 2.6: Visualization of filters learned by RBMs with 36 hidden neurons on MNIST and 1-MNIST after 5 epochs using traditional learning algorithms.

1) *Learning rate scheduling*: Due to its stochastic nature (when only part of data, i.e. mini-batch, is used to compute the gradient), the gradient does not tend to approach zero. Therefore, the learning rate is typically forced towards zero at the end of training. However, if the learning rate is annealed too quickly, then the RBM will not learn anything, but only stay in the plateau of the learning space where most of the weights stay close to zero.

2) *Weight decay* prior regularizes the indefinite growth of the norm of the parameters, which sometimes happens in practice. This yields the following update rules:

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

3) *Momentum* is used to smoothen the gradients yielding a modified update rule:

$$w_{ij} \leftarrow w_{ij} + \eta \left[(1-\beta) \nabla_{w_{ij},t} + \beta \nabla_{w_{ij},t-1} \right],$$

where $\nabla_{w_{ij},t}$ and $\nabla_{w_{ij},t-1}$ are the gradients computed at the current and previous iterations and $0 \leq \beta < 1$ is a momentum parameter.

4) In order to avoid having meaningless hidden neurons, there have been attempts to *sparsify* the activations of the hidden neurons (Hinton, 2010; Lee et al., 2008). The sparsity can be achieved by adding a regularization term that penalizes a deviation of the expected

activation from a fixed level p:

$$\rho \sum_j |p - \langle h_j \rangle_{P_0}|^2,$$

where ρ denotes a degree of regularization.

The proposed heuristics are known to help in many practical applications. However, they all introduce 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). The stochastic gradient learning of RBM can easily diverge even when the proposed heuristics are used, if the associated parameters are not chosen carefully (Schulz et al., 2010; Fischer & Igel, 2010).

Chapter 3

Parallel Tempering Learning

While contrastive divergence learning has been considered an efficient way to learn RBM , it has a drawback due to a biased approximation in the learning gradient. This chapter proposes to use an advanced Monte Carlo method called parallel tempering instead, and shows experimentally that it works efficiently. A part of the work described in this chapter was reported in the author's paper (Cho et al., 2010).

3.1 Parallel Tempering and Restricted Boltzmann Machines

Training RBMs using the stochastic gradient updates requires that it must be possible to efficiently sample from the data distribution $P(\mathbf{h} \mid \mathbf{v}, \boldsymbol{\theta})$ and the model distribution $P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})$. Thanks to the simple structure and formulation of BM and RBM , a Gibbs sampling is enough to obtain the samples. However, its inefficiency led to the contrastive divergence (CD) learning and its variants which do not follow the exact gradient, but rather, approximate the exact gradient. Its nature of simplicity and computational efficiency made the CD learning huge success in training RBMs, but still the CD learning has disadvantages. For more detailed discussion on the topic, Chapter 2 should be referred.

A problem that has not been addressed neither by Gibbs sampling nor by CD learning is that the samples generated during the negative phase do not tend to explain the whole state space. This section, therefore, proposes to use another improved variant of Markov-Chain Monte Carlo sampling method called *parallel tempering* (PT).

3.1.1 Parallel Tempering

The introduction of PT sampling goes back to 1980s when Swendsen & Wang (1986) introduced a replica Monte Carlo simulation and applied it to the Ising model which is equivalent to a Boltzmann machine with only visible neurons. The replica Monte Carlo simulation proposed to simulate multiple copies of particles (replica) with different temperature concurrently rather than simulating them sequentially. Similarly, Geyer (1991) later presented applying parallel chaining of MCMC sampling based on the *speed* of mixing of samples across parallel chains to the maximum likelihood estimator.

Afterward, there have been many approaches of applying parallel tempering to other fields. Those fields include the simulations of polymers, proteins, and states of solid materials, and even, studies of phase transitions at the quantum levels (for more applications, see Earl & Deem, 2005).

In the rest of this section, PT sampling having multiple Gibbs sampling chains with varying levels of temperatures used to obtain *good* samples from the state space is briefly discussed.

The basic idea of PT sampling is that samples are collected from multiple chains of Gibbs sampling with different temperatures[1]. The term *temperature* in this context denotes the level of the energy of the overall system. The higher the temperature of the chain, the more likely the samples collected by Gibbs sampling move freely.

For every pair of collected samples from two distinct chains, the swap probability is computed, and the samples are swapped according to the probability. The swap probability of a pair of samples is formulated according to the Metropolis rule (see, e.g., Mackay, 2002) as

$$P_{\text{swap}}(\mathbf{x}_{T_1}, \mathbf{x}_{T_2}) = \min\left(1, \frac{P_{T_1}(\mathbf{x}_{T_2})P_{T_2}(\mathbf{x}_{T_1})}{P_{T_1}(\mathbf{x}_{T_1})P_{T_2}(\mathbf{x}_{T_2})}\right), \tag{3.1}$$

where T_1 and T_2 denote the temperatures of the two chains, and $\mathbf{x}_{T_1}$ and $\mathbf{x}_{T_2}$ denote samples collected from the two chains.

After each round of sampling and swapping, the sample at the true temperature $T = 1$ is gathered as the sample for the iteration. The samples come from the true distribution , $P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})$ in case of RBMs, assuming that enough iterations are run to diminish the effect of the initialization.

It must be noted that the Gibbs sampling chain with the highest temperature ($T = 0$) is never

[1]Since the lower value denotes the higher temperature, a term *inverse temperatures* from the highest temperature $T = 0$ to the current temperature $T = 1$ is frequently used, but in this thesis, *temperature* will be used.

multi-modal such that all the neurons are mutually independent and likely to be active with probability $\frac{1}{2}$. So, the samples from the chain are less prone to missing some modes. From the chain with the highest temperature to the lowest temperature, samples from each chain become more and more likely to follow the target model distribution. How PT sampling could avoid being trapped into a single mode is illustrated in Figure 3.1.

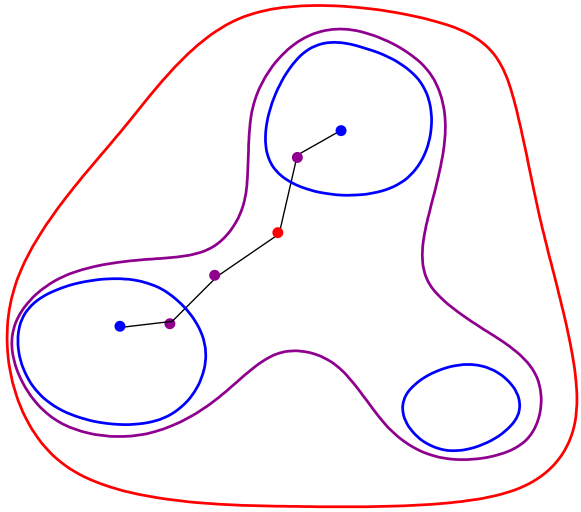


Figure 3.1: Illustration of how PT sampling could avoid being trapped in a single mode. The red, purple, and blue curves and dots indicate distributions and the samples from the distributions with the high, medium, and cold temperatures, respectively. Each black line indicates a single sampling step.

This nature of swapping samples between the different temperatures enables better mixing of samples from different modes with much less number of samples than that would have been required if Gibbs sampling was used.

3.1.2 Parallel Tempering Learning

PT sampling in training RBMs can be simply uses as a replacement of Gibbs sampling in the negative phase. This method is, from now on, referred to as PT learning. Due to the previously mentioned characteristics, it is expected that the samples collected during the negative phase would explain the model distribution better, and that the learning process would be successful even with a smaller number of samples than those required if Gibbs sampling is used.

A brief description of how PT sampling can be carried out for RBMs is given in Algorithm 3. This is the procedure that is run between each parameter update during learning.

Algorithm 3 PT sampling steps for an RBM

Create a sequence of RBMs $(R_0, R_1, \cdots, R_K)$ such that parameters of R_k are $\boldsymbol{\theta}_k = (T_k\mathbf{W}, T_k\mathbf{b}, T_k\mathbf{c})$, where $0 \leq T_0 < T_1 < \cdots < T_K = 1$.
Create an empty set of samples $\boldsymbol{X} = \{\}$.
Set $\mathbf{x}_0 = (\mathbf{x}_{0,0}, \cdots, \mathbf{x}_{K,0})$ such that every $\mathbf{x}_{k,0}$ is a uniformly distributed random vector (or use old ones from the previous epoch).
for $m = 1 \cdots M$ **do**
  Sample $\mathbf{x}_m = (\mathbf{x}_{0,m}, \cdots, \mathbf{x}_{K,m})$ from the sequence of RBMs such that $\mathbf{x}_{k,m}$ is sampled by one-step Gibbs sampling starting from $\mathbf{x}_{k,m-1}$.
  for $j = 2 \cdots K$ **do**
    Swap $\mathbf{x}_{j,m}$ and $\mathbf{x}_{j-1,m}$ according to $P_{\text{swap}}(\mathbf{x}_{j,m}, \mathbf{x}_{j-1,m})$ computed using (3.1).
  end for
  Add $\mathbf{x}_{K,m}$ to $\boldsymbol{X}$.
end for
$\boldsymbol{X}$ is the set of samples collected by parallel tempering sampling.

3.2 Experiments

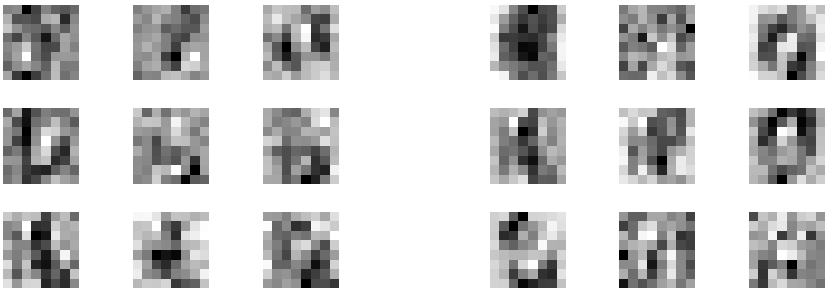
Two different sets of experiments were made. The goal of the first set of experiments was to test the capability of RBMs to capture the data distribution. Samples were generated from the RBM trained on the OptDigits data set. The data set was acquired from the UCI Machine Learning Repository (Asuncion & Newman, 2007) and it consisted of handwritten digits of the size 8×8 pixels. The samples were collected by parallel tempering sampling starting from a randomly drawn state. Most of the samples were observed to resemble the digits regardless of the initial state.

The second set of experiments was conducted in order to compare the performance of RBMs depending on two different learning methods: CD learning and learning using sampling with PT. The performance was evaluated by the estimated likelihood of the training data set and the estimated probability of the test data set, both computed by annealed importance sampling (AIS).

Furthermore, in the second experiment the probability of uniformly randomly generated data is computed for the current RBM model. The goal was to observe a potential problem of CD learning that the samples generated during the negative phase do not represent the state space as well as the samples generated by PT sampling, but only represent the region centered around the training samples (Bengio, 2009). The probability of random data was computed for different learning methods and compared.[2]

[2]We assume that uniformly drawn samples do not lie close to the training data because the size of the training data set is much smaller than the size of the state space which is 2^{64}.

(a) Training data set



(b) Visualization of hidden nodes (CD1) (c) Visualization of hidden nodes (PT)

Figure 3.2: Training data set and visualization of hidden nodes. (a): 10 training samples where for each digit one sample was randomly chosen. (b) (c): the weights associated with nine randomly chosen hidden neurons.

3.2.1 Generating samples from a trained restricted Boltzmann machine

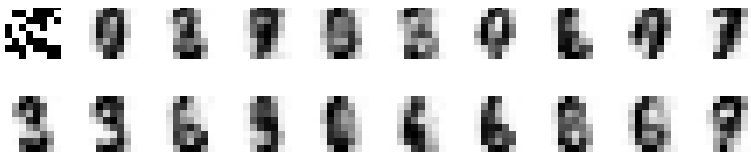
RBMs were constructed such that there are 64 visible neurons and 100 hidden neurons. Each RBM was trained with 3822 training samples of 8×8 handwritten digits. The original OptDigits data set provides 17-level greyscale digits, but for simplicity the intensity of each pixel was rounded so that the intensity less than 8 became 0 (and 1 otherwise).

Each RBM was trained separately by CD learning with $n = 1$ and learning with PT sampling. PT sampling was done with $K = 20$ temperatures $T_0 = 0$, $T_1 = 0.05, \ldots, T_{20} = 1$. The models represented by the RBMs are named CD1 and PT, respectively. Each gradient update was done in the full-batch style so that all the training samples were used. CD1 and PT were trained for 2000 epochs, and the learning rate η started from 0.05 and gradually decreased following the search-then-converge scheduling such that the learning rate $\eta(t)$ at the t-th update is

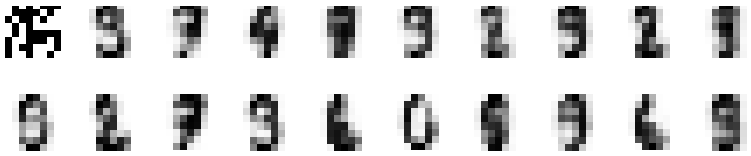
$$\eta(t) = \frac{\eta(0)}{1 + \frac{t}{t_0}},$$

where $\eta(0) = 0.05$ for both the weight and the bias, and $t_0 = 300$ for both CD1 and PT.

Figure 3.2 shows the training data samples and the visualization of the hidden nodes after training. The visualization of the hidden node was done by displaying the weights associ-



(a) CD1



(b) PT

Figure 3.3: Samples generated by parallel tempering sampling from the RBM trained with (a) CD1 and (b) PT started from the random sample. The first digits of both figures are the random initial samples.

ated with the node as a grey-scale digit. It can be observed that each hidden node represents a distinct feature.

Figure 3.3 shows the activation probabilities for the visible neurons of the generated samples from the models learned with CD1 and PT. The digits in the figure are 19 samples chosen out of 2000 samples collected by PT sampling starting from the random sample. Each consecutive samples are separated by 100 sampling steps, and the first digit in both figures of Figure 3.3 represents the random initial sample. It is clear that the trained RBM is able to generate digits which look similar to the training data regardless of the training methods.

3.2.2 Comparison between CD learning and PT learning

For the second experiment, RBMs with 100 hidden neurons were trained using four learning algorithms: CD1, CD5, CD25 and PT, where CDn denotes CD learning with n Gibbs sampling steps per gradient update.

The parameters K and M of parallel tempering were chosen so that the number of total Gibbs sampling steps during one gradient update matches that of CD1 which uses as many samples as the number of the training data samples. PT was, therefore, trained with $K = 20$ temperatures and $M = 192$ samples per gradient update. This choice is reasonable in the sense that the difference in CD learning and learning with PT sampling only depends on the number of Gibbs sampling steps, whereas the computational cost of additional operations may vary largely depending on the implementation.

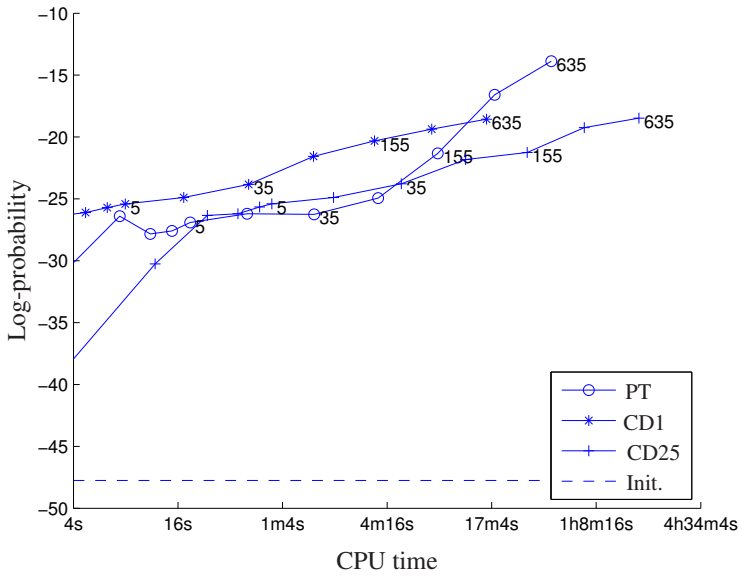


Figure 3.4: Average probabilities of test data against the processor time. The dashed line indicates the initial log-likelihood. The numbers denote the number of epochs after which the value was measured.

Each RBM was trained for 635 epochs and the probabilities of both training and test data were estimated. 50 AIS runs with 5000 temperatures were averaged to obtain the estimate of the normalizing constants. All the models were trained 30 times and the averaged performance indices were calculated.

Figure 3.4 and Figure 3.5 show that the probability of the test data and the likelihood of the model increase, while the probability of the random data decreases over the gradient updates. This is consistent with the fact that the gradient maximizes the likelihood according to the distribution of the training data. Figure 3.6 confirms that the probability of the unseen samples that are not close to any training sample is decreased.

However, the rate of the changes in the likelihood and the probability of the test data over updates differs from one model to another. PT achieves the highest average likelihood and the highest average probability of the test data, and at the same time achieves the lowest probability of the random data at the fastest rate. It can be further observed that PT learning is computationally more favorable than CD25 and comparable to CD1.

Figure 3.7 shows the average probability of the test data set and the random data set by 30 independent trials. These results confirm that PT indeed achieves the highest probability of the test data set and the lowest probability of the random data set. It should be, however, noted that the variance of PT is greater than those of both CD1 and CD25.

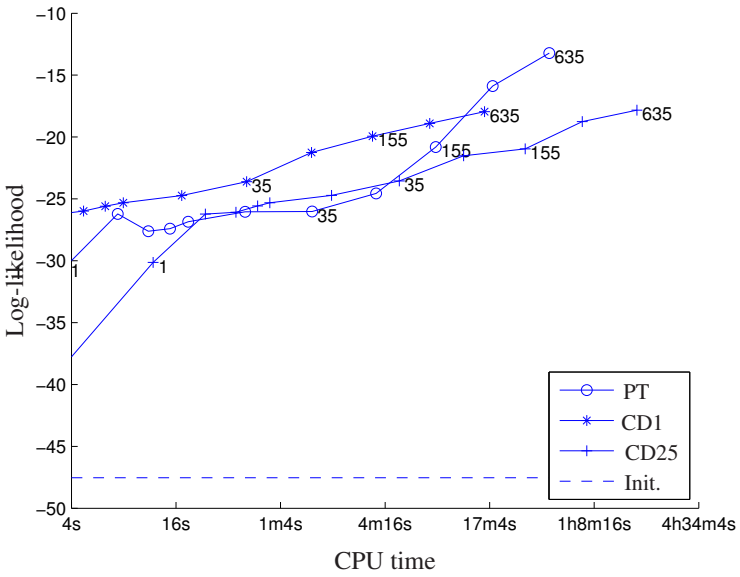


Figure 3.5: Average log-likelihood of the model against the processor time. The dashed line indicates the initial log-probability of test data. The numbers denote the number of epochs after which the value was measured.

The increase of n in CD learning certainly boosts up the rate of the increase in the likelihood as a function of learning epochs, but even with $n = 25$ CD learning cannot achieve as large likelihood as PT does. CD learning with $n = 25$ is much more computationally demanding than PT. This result indicates that the use of the advanced sampling technique can yield faster and better training of RBMs.

3.3 Practical Consideration

Although the experiments showed that gathering enough number of samples from a single PT sampling chain consisting of multiple parallel Gibbs sampling chain is sufficient to train RBMs, PT learning showed its weakness evidenced by the large variance of the resulting RBMs with different initializations.

In order to reduce the high variabilities in training RBMs, PT learning can borrow the idea from CD and PCD learning introduced in Chapter 2 such that there are multiple sets of multiple Gibbs sampling chains starting from the training samples in the initial minibatch, or full-batch. For each update, n steps of PT sampling is performed starting from the model samples obtained in the previous update. For every n_{swap} updates, the swapping of the

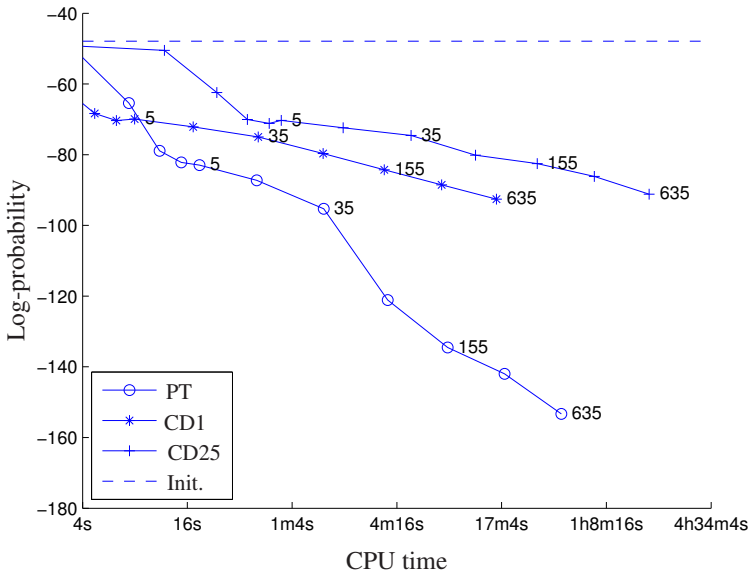


Figure 3.6: Average probabilities of random data against the processor time. The dashed line indicates the initial log-probability of random data. The numbers denote the number of epochs after which the value was measured.

samples can be performed, where n_{swap} is a small positive integer[3].

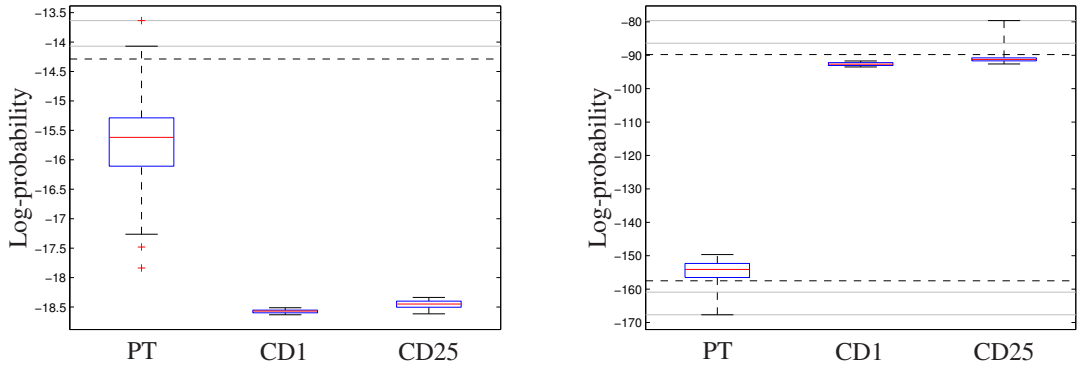
Although the experimental results with this modification are not presented in this chapter, it is clear from the experiments in Chapter 4 and Chapter 5 that the variance is reduced significantly when the modification is employed.

3.4 Conclusions

This chapter proposed an alternative approach which utilizes parallel tempering for training RBMs. This approach does not sacrifice the optimality of the direction of the gradient, as CD learning does, but reduces the computational cost by improving the quality of the samples.

Two separate experiments were done for (1) confirming the capability of RBM to capture the data distribution and (2) showing that RBM trained by the proposed PT approach is superior to that trained by the conventional CD learning. The former experiment confirmed that RBM trained by either CD learning or learning with PT sampling is able to generate samples resembling the training data. The second experiment confirmed that the use of the proposed PT approach can result in a more accurate RBM. As a performance measure, the

[3]This practical modification is mathematically justified by Younes (1989).



Test data Random data

Figure 3.7: Left: Box plots of probabilities of test data after 635 epochs over 30 repeated runs. Right: Box plots of probabilities of random data over 30 repeated runs. For probability values, the red line inside the box denotes the median, the edges of the box are 25-th and 75-th percentiles, and the whiskers are extended to the extreme value not considering possible outliers.

log-likelihood estimated using AIS was used.

Learning with PT sampling was superior in all aspects of the experimental results. We observed higher likelihood computed on the training data and higher probability of the test data. The increase of the likelihood over the gradient updates was also faster. The probability of random samples by PT sampling was less than any other model trained with CD learning. This confirmed the existence of the potential problem of CD learning that the samples generated by CD learning during the negative phase do not represent the state space well and fail to decrease the probabilities over the regions which are far from the training data. At the same time, the computational complexity of the gradient update by PT sampling was comparable to that of CD learning.

Recently, the use of PT learning for RBMs has been proposed independently also by Desjardins et al. (2010b). Desjardins et al. (2010b) illustrated the possible explanations why PT learning performs better than CD learning, and presented the experimental results showing the superiority of PT learning. This chapter essentially showed the similar results, and additionally showed that PT learning could be as efficient as CD learning in terms of the computational complexity.

Additionally, Desjardins et al. (2010a) proposed an adaptive method for maintaining the optimal distributions with different temperatures. The method shows its superiority compared to PCD learning and PT learning with the fixed configuration of temperatures. However, the computational cost of the method is higher than maintaining the fixed number of tempered distributions.

Chapter 4

Enhanced Gradient and Adaptive Learning Rate

In this chapter, a new training algorithm which addresses the difficulties of training RBMs discussed in Chapter 2 is proposed. 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.

Extensive experiments comparing the conventional learning algorithms with the proposed one are presented at the end of the chapter. The experiments 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.

As discussed previously in Chapter 2, the data set 1-MNIST is known to be more difficult to learn, and the chapter gives an explanation fot this effect. The empirical results suggest that the new learning rules can avoid many difficulties in training RBMs including learning dense data sets.

4.1 Adaptive Learning Rate

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

Algorithm 4 Adaptive Learning Rate

Input: parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$
previous learning rate η_0
gradients $\mathbf{G} = (\nabla_{\mathbf{W}}, \nabla_{\mathbf{b}}, \nabla_{\mathbf{c}})$
data samples $\mathbf{X}_{\mathrm{d}}$
model samples $\mathbf{X}_{\mathrm{m}}$
Prepare a set of candidate learning rates C_η based on η_0.
for η in C_η **do**
  Compute a candidate model $\boldsymbol{\theta}'$ by (2.9) – (2.11).
  Compute a local likelihood $P_{\boldsymbol{\theta}'}(\mathbf{v}_{\mathrm{d}})$ in (4.1)
end for
$\tilde{\eta} = \arg\max_\eta c_\eta$.
Output: learning rate $\tilde{\eta}$

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 η and $P_{\boldsymbol{\theta}}(\mathbf{v}) = P^*_{\boldsymbol{\theta}}(\mathbf{v})/Z_{\boldsymbol{\theta}}$ is the probabilistic density function (pdf) with normalizing constant $Z_{\boldsymbol{\theta}}$ for the model with parameters $\boldsymbol{\theta}$. Assuming 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 (2.13):

$$P_{\boldsymbol{\theta}'}(\mathbf{v}_{\mathrm{d}}) = \frac{P^*_{\boldsymbol{\theta}'}(\mathbf{v}_{\mathrm{d}})}{Z_{\boldsymbol{\theta}}} \frac{Z_{\boldsymbol{\theta}}}{Z_{\boldsymbol{\theta}'}} = \frac{P^*_{\boldsymbol{\theta}'}(\mathbf{v}_{\mathrm{d}})}{Z_{\boldsymbol{\theta}}} \left\langle \frac{P^*_{\boldsymbol{\theta}'}(\mathbf{v})}{P^*_{\boldsymbol{\theta}}(\mathbf{v})} \right\rangle^{-1}_{P_{\boldsymbol{\theta}}}, \tag{4.1}$$

where $\mathbf{v}_{\mathrm{d}}$ denotes the training data.

Now a learning rate is selected so as to maximize the likelihood of the new parameters $\boldsymbol{\theta}'$. Equation (4.1) can be used to approximate the required likelihood. The unnormalized pdf $P^*_{\boldsymbol{\theta}'}$ is computed using the training samples and (2.8), 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 the gradients and therefore computing this expectation can be done practically for free [1].

The pseudo-code for the adaptive learning rate is provided in Algorithm 4.

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 of the mini-batch. In our experiments, the new learning rate was selected from the set $\{(1-\epsilon)^2\eta_0, (1-\epsilon)\eta_0, \eta_0, (1+\epsilon)\eta_0, (1+\epsilon)^2\eta_0\}$, where η_0 is the previous learning rate and ϵ is a small constant.

[1] The 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. Hence, in practice, it is advised to use samples from the next mini-batch.

4.2 Enhanced Gradient

In this section, a new gradient is proposed to be used instead of (2.9)–(2.11). Let the covariance between two variables under distribution P be defined as:

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

Then, the standard gradient (2.9) can be rewritten as

$$\nabla w_{ij} = \mathrm{Cov}_{\mathrm{d}}\left(v_i, h_j\right) - \mathrm{Cov}_{\mathrm{m}}\left(v_i, h_j\right) + \left\langle v_i \right\rangle_{\mathrm{dm}} \nabla c_j + \left\langle h_j \right\rangle_{\mathrm{dm}} \nabla b_i \,, \tag{4.2}$$

where ∇c_j and ∇b_i are the gradients defined in (2.10)–(2.11) and $\left\langle \cdot \right\rangle_{\mathrm{dm}} = \frac{1}{2}\left\langle \cdot \right\rangle_{\mathrm{d}} + \frac{1}{2}\left\langle \cdot \right\rangle_{\mathrm{m}}$ is the average activity of neuron under the data and model distributions.

The standard gradient (4.2) has several potential problems. The gradients with respect to the weights contain the terms that point to the same direction as the gradient with respect to the bias terms (and vice versa). This effect is prominent when there are many neurons which are mainly active, that is for which $\left\langle x_i \right\rangle_{\mathrm{dm}} \approx 1$. 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 Figure 2.6(b). When $\left\langle x_i \right\rangle_{\mathrm{dm}} \approx 0$ 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 discussed in Section 2.4.3, have been successful.

A related problem is that the update using (4.2) 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:

$$\tilde{x}_k = x_k^{1-f_k}(1 - x_k)^{f_k} \,, \qquad f_k \in \{0, 1\} \,, \tag{4.3}$$

where x_k can be either a visible or a hidden neuron. The parameters can then be transformed accordingly to $\tilde{\boldsymbol{\theta}}$:

$$\tilde{w}_{ij} = (-1)^{f_i + f_j} w_{ij}, \tag{4.4}$$

$$\tilde{b}_i = (-1)^{f_i}\left(b_i + \sum_j f_j w_{ij}\right), \tag{4.5}$$

$$\tilde{c}_j = (-1)^{f_j}\left(c_j + \sum_i f_i w_{ij}\right), \tag{4.6}$$

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 $\mathbf{x}$. Details on obtaining the transformations are described in Appendix B.1.

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

$$\begin{aligned} w_{ij} &\leftarrow w_{ij} + \eta\Big[\langle v_i h_j \rangle_{\mathrm{d}} - \langle v_i h_j \rangle_{\mathrm{m}} - f_i \left(\langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}} \right) - f_j \left(\langle v_i \rangle_{\mathrm{d}} - \langle v_i \rangle_{\mathrm{m}} \right) \Big] \\ &= w_{ij} + \eta\Big[\mathrm{Cov}_{\mathrm{d}} \left(v_i, h_j \right) - \mathrm{Cov}_{\mathrm{m}} \left(v_i, h_j \right) \\ &\qquad + \left(\langle v_i \rangle_{\mathrm{dm}} - f_i \right) \nabla c_j + \left(\langle h_j \rangle_{\mathrm{dm}} - f_j \right) \nabla b_i \Big] \end{aligned} \tag{4.7}$$

$$b_i \leftarrow b_i + \eta\Big[\nabla b_i - \sum_j f_j \left(\nabla w_{ij} - f_i \nabla c_j - f_j \nabla b_i \right) \Big] \tag{4.8}$$

$$c_j \leftarrow c_j + \eta\Big[\nabla c_j - \sum_i f_i \left(\nabla w_{ij} - f_i \nabla c_j - f_j \nabla b_i \right) \Big], \tag{4.9}$$

where $\nabla \boldsymbol{\theta}$ are the gradients defined in Eqs. (2.9)–(2.11). More details on the derivations are provided in Appendix B.2.

There are, thus, $2^{n_v+n_h}$ different update rules defined by different combinations of binary f_k, $k = 1, \ldots, n_v+n_h$, 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.

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

$$\prod_{k=1}^{n_v+n_h} \langle x_k \rangle_{\mathrm{dm}}^{f_k} \left(1 - \langle x_k \rangle_{\mathrm{dm}} \right)^{1-f_k} . \tag{4.10}$$

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

$$\tilde{\nabla} w_{ij} = \mathrm{Cov}_{\mathrm{d}} \left(v_i, h_j \right) - \mathrm{Cov}_{\mathrm{m}} \left(v_i, h_j \right) \tag{4.11}$$

$$\tilde{\nabla} b_i = \langle v_i \rangle_{\mathrm{d}} - \langle v_i \rangle_{\mathrm{m}} - \sum_j \langle h_j \rangle_{\mathrm{dm}} \tilde{\nabla} w_{ij} \tag{4.12}$$

$$\tilde{\nabla} c_j = \langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}} - \sum_i \langle v_i \rangle_{\mathrm{dm}} \tilde{\nabla} w_{ij}, \tag{4.13}$$

in which, by the choice of the weights (4.10), the effect of the bias gradients in ∇w_{ij} is canceled out completely. In Appendix B.3, detailed derivations on how the weighted sums

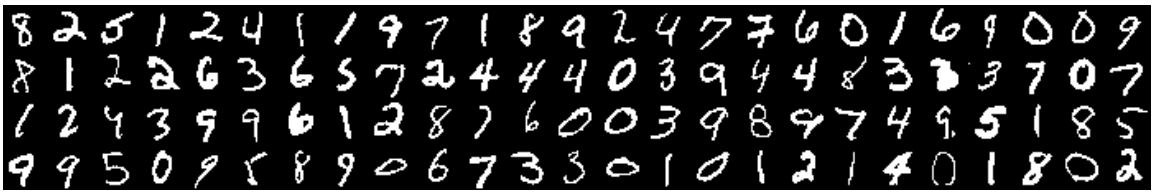


Figure 4.1: 100 randomly chosen samples from MNIST data set.

are computed are provided.

The new rules are invariant to the bit-flipping transformations. One can note that the enhanced gradient shares all zeroes with the traditional gradient.

In Figures 4.2–4.3, some experimental analysis of the proposed gradient is presented. Figure 4.2 shows the norms of the gradient for the weights of the RBM with 361 hidden neurons trained on MNIST data set (see Figure 4.1 for examples of samples). It is clear that the additional terms that distract learning dominate in the traditional gradient, especially at the early stage of training.

Figure 4.3 shows the differences in the update directions for different neurons of the RBM trained on MNIST. Each element of a matrix is the absolute value of the cosine of the angle $c(\cdot, \cdot)$ between the update directions for the two neurons, where the cosine of the angle between two vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$c(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}.$$

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.

4.3 Experiments

In this section, the proposed improvements and the traditional learning algorithms are experimentally compared. In Sections 4.3.1–4.3.3, RBMs are trained on the MNIST data set (LeCun et al., 1998), and in Section 4.3.4, the Caltech 101 Silhouettes data (Marlin et al., 2010) is used.

All experiments run 20 epochs with a mini-batch size of 128 unless otherwise mentioned.

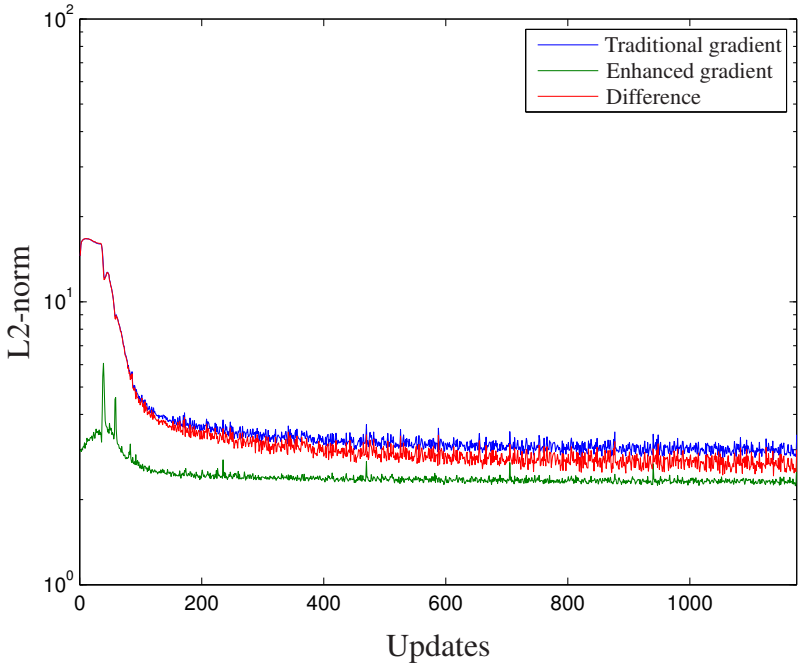


Figure 4.2: L2-norms of the gradients for weights while training RBM with 361 hidden neurons. The blue curve plots the norms of the traditional gradient, and the green curve plots the norms of the proposed robust gradient. The norms of the difference between two gradients are drawn with the red curve.

Thus, each RBM was updated about 9,400 times. Both biases $\mathbf{b}$ and $\mathbf{c}$ were initialized to all zeros. Weights were randomly initialized such that $w_{ij} = \lambda \cdot u$ where λ is a weight scale and $u \sim \mathcal{U}(-1, 1)$ denotes a sample from the uniform distribution on -1 to 1. By default, $\lambda = 1/\sqrt{n_v + n_h}$ was used.

For PT learning, there were 11 different temperatures equally spaced from $t_0 = 0$ to $t_{10} = 1$. For CD learning, each update performed $n = 1$ step 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. Each AIS used RBMs with parameters $\boldsymbol{\theta}_i = (t_i\mathbf{W}, t_i\mathbf{b}, t_i\mathbf{c})$ and 10,001 equally-spaced temperatures from 0 to 1. Each estimate of $Z(\boldsymbol{\theta})$ was averaged over 100 independent AIS runs.

4.3.1 Sensitivity to Learning Rate

In order to demonstrate how the learning rate can greatly affect training results, RBMs with 361 hidden neurons were trained using the traditional gradient with five different learning

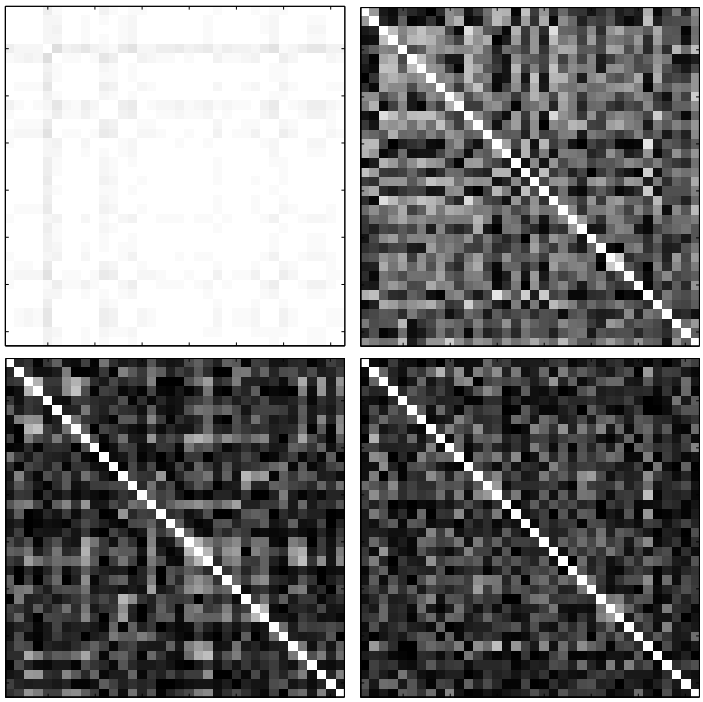


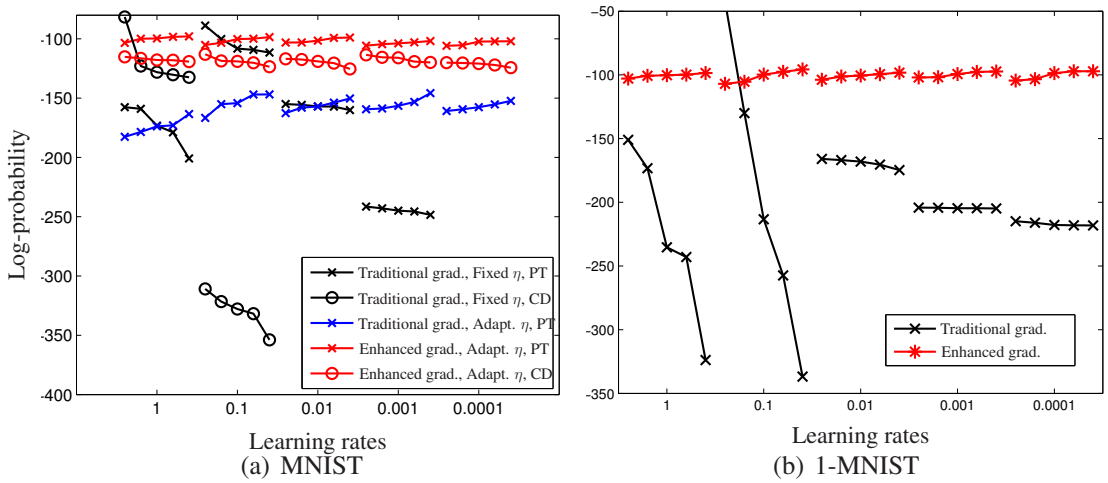
Figure 4.3: The angles between the update directions for the weights of the RBM with 36 hidden neurons. White pixels correspond to small angles, while black pixels correspond to orthogonal directions. From left to right: (top) traditional gradient after 26 updates, traditional gradient after 352 updates, (bottom) enhanced gradient after 26 updates, and enhanced gradient after 352 updates.

rates $\{1, 0.1, 0.01, 0.001, 0.0001\}$.

The black curves in Figure 4.4(a) 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 the 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, RBMs with 361 hidden neurons were trained 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 Figure 4.4(a) show the obtained log-probabilities of the test data. The results are now more stable and the variance among the resulting RBMs trained with different initial learning rates 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 the RBMs were trained quite well.

These results suggest that the adaptive learning rate works well. However, it was still



(a) MNIST (b) 1-MNIST

Figure 4.4: Log-probabilities of test data samples computed after 20 epochs for five runs with different initializations for different learning rates on MNIST (left) and 1-MNIST (right). Log-probabilities that do not appear on the plot are smaller than -400. In case of 1-MNIST, only the results obtained using PT learning are shown.

slightly better to use manually tuned training parameters (using the constant learning rate of 0.1).

Figure 4.5 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 Figure 4.4(a) show the log-probabilities of the test data obtained with the new enhanced 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 which is shown in Figure 4.4(b). For 1-MNIST, only PT learning was used for comparing the traditional learning method and the proposed one. It is clear that the traditional gradient with the fixed learning rate results in huge variance depending on both the learning rate and the initialization of the parameters[2]. On the other hand, the proposed method combining the enhanced gradient with the adaptive learning rate provides the consistent result, regardlessly. This confirms that the new enhanced gradient is invariant to data representation.

[2] A higher log-probability obtained using the traditional learning method with the learning rate fixed to 0.1 may be due to the failure of AIS (see, Schulz et al., 2010, for examples of the failure of AIS).

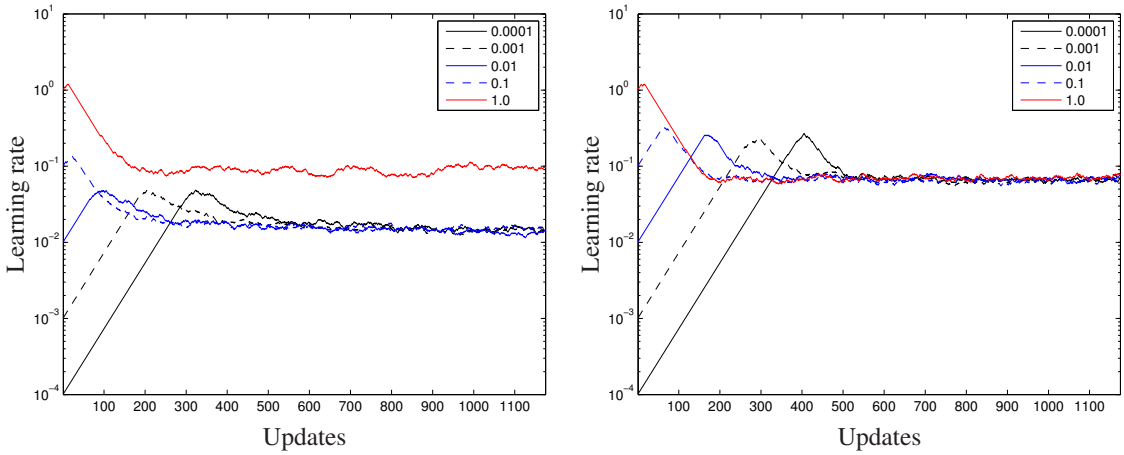


Figure 4.5: Evolution of the adaptive learning rate from five different initializations during learning. The learning rates are shown as a function of the number of updates. The RBMs were trained with the traditional gradient (left) and the enhanced gradient (right).

4.3.2 RBM as Feature Extractor

In addition to the log-probabilities of the test data, simple logistic regression classifiers were trained 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; Ciresan et al., 2010). However, the relative difference in the accuracies between two rules can be used as a guide for assessing the superiority of the proposed method.

The black curves in Figure 4.6 show high variance of the classification results for the traditional gradient depending on the chosen learning rate. The results obtained for MNIST (the left plot) are pretty good although the choice of the learning does have an effect on performance. However, the classification accuracy obtained for 1-MNIST (the right plot) is very bad, which proves that 1-MNIST is more difficult for learning using the traditional gradient.

The blue curves in Figure 4.6 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 Figure 4.6 shows the superior performance of the enhanced gradient and the adaptive learning rate compared to the traditional gradient. Regardless of the initial

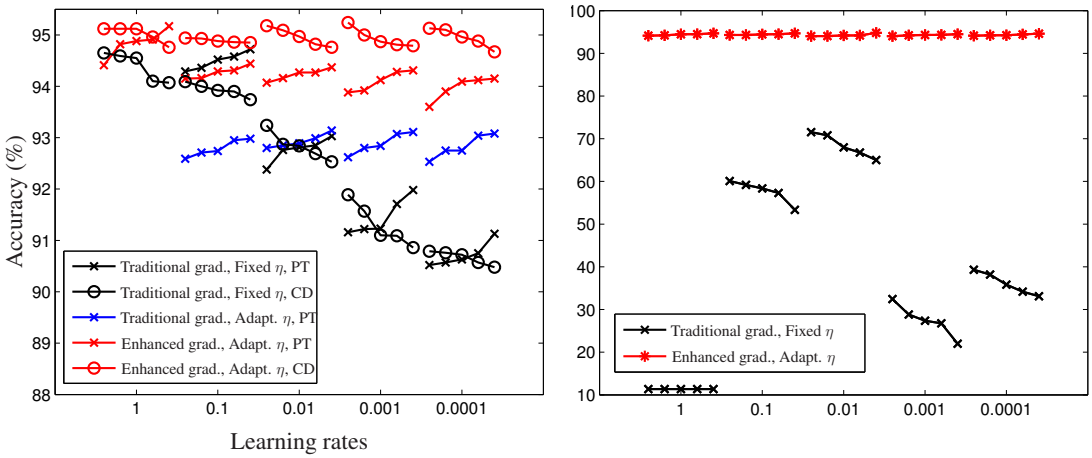


Figure 4.6: Classification accuracy of test data samples computed after 20 epochs for MNIST (left) and 1-MNIST (right). For each initial learning rate, the learning was conducted five times. The results that do not appear on the upper plot were below 88% for MNIST and below 10% for 1-MNIST. For 1-MNIST, only the results obtained using PT learning are shown.

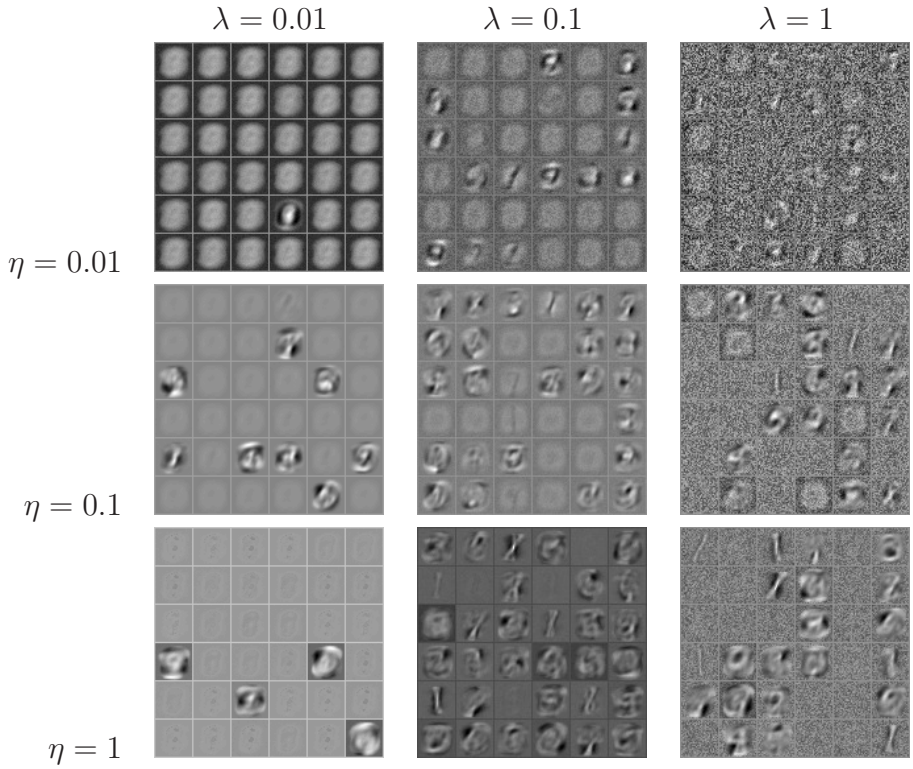
learning rate, all the RBMs leaned features which yielded high classification performance. Note that the results are excellent also for 1-MNIST.

4.3.3 Sensitivity to Weight Initialization

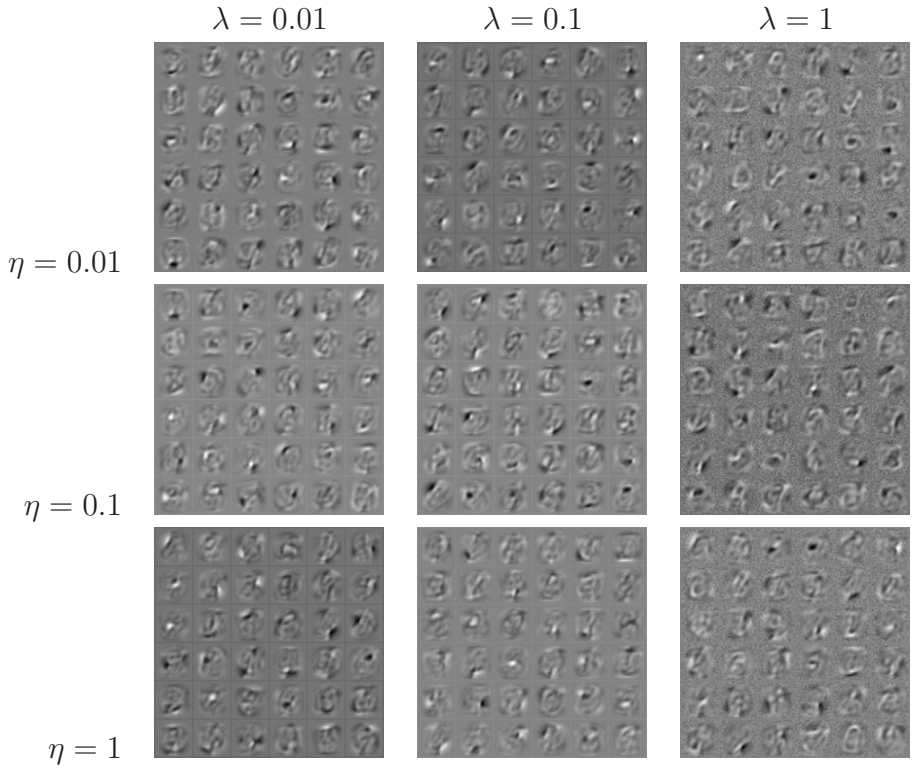
In the next experiment, the sensitivity of training results to the scale λ of the weight initialization is investigated. Small RBMs with 36 hidden neurons were trained on MNIST using different scales of the initial weights and varying learning rates. Here, PT sampling was used to draw model samples from the RBMs.

Figure 4.7(a) visualizes the filters learned by the 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.

Figure 4.7(b) 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.



(a) Traditional gradient with fixed η



(b) Enhanced gradient with adaptive η

Figure 4.7: Visualization of filters learned by RBMs with 36 hidden neurons on MNIST with various initial learning rates and initial weights scaling. Learning was performed for 5 epochs each.

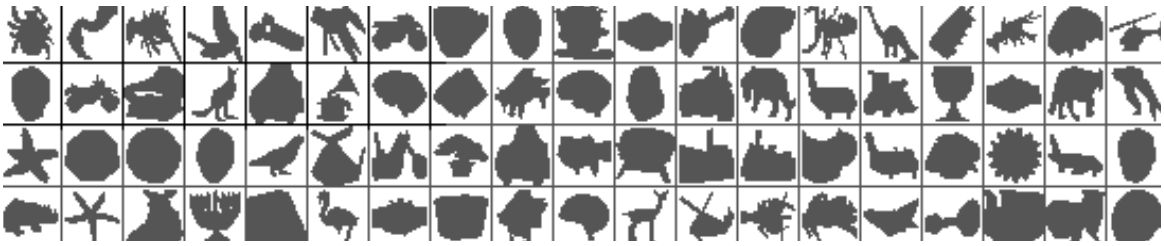


Figure 4.8: 100 randomly chosen samples from Caltech 101 Silhouettes data set.

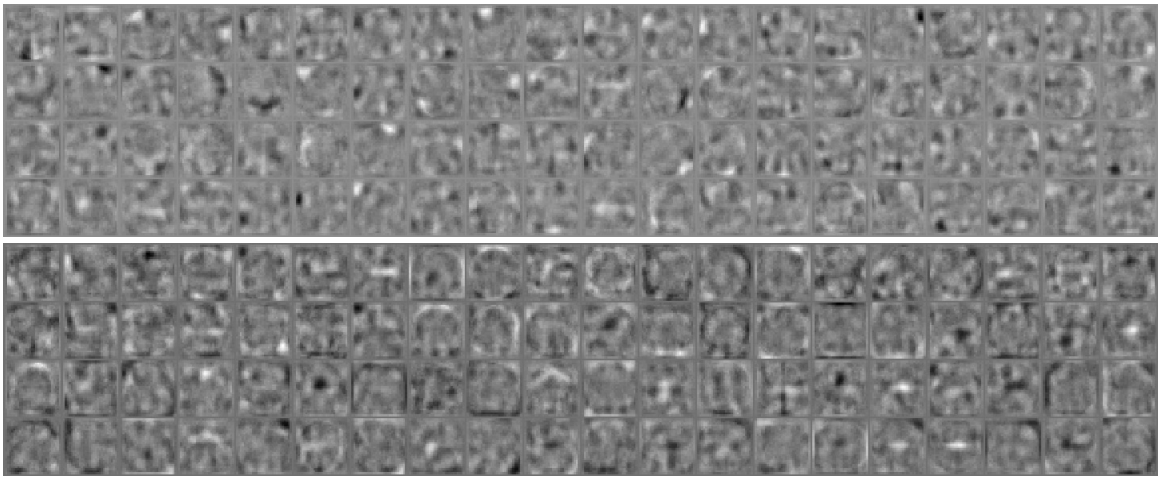


Figure 4.9: Visualization of weights learned by an RBM with 1000 hidden neurons on Caltech 101 Silhouettes data set. (Top) 80 filters with large L2-norms. (Bottom) 80 filters with small L2-norms.

4.3.4 Caltech 101 Silhouettes

Finally, the proposed learning rules were experimented on Caltech 101 Silhouettes data set (Marlin et al., 2010). Figure 4.8 presents randomly chosen samples from 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.

With the hidden activations obtained from the trained RBMs, simple logistic regression classifiers were trained to check the classification accuracies.

The obtained results are presented in Table 4.1. Remarkably, the classification accuracy improved by more than 5 % over the best result reported by Marlin et al. (2010).

Figure 4.9 shows two sets of 100 filters of the RBM with 1000 hidden neurons that have the largest L2-norms (left) and the least L2-norms (right), respectively.

Hidden neurons	Log-probability		Accuracy (%)	
	PT	CD	PT	CD
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 4.1: Log-probabilities and classification accuracies of the test data of Caltech 101 Silhouettes after 300 epochs.

4.4 Conclusions

This chapter experimentally showed the difficulties of training RBMs discussed previously in Chapter 2, and then, a new algorithm for training RBMs that addresses those difficulties was proposed. It consists of an adaptive learning rate and an enhanced gradient, and is formulated with well-founded theoretical background. The proposed algorithm was experimented extensively with RBMs on the MNIST handwritten digits and Caltech 101 Silhouettes data set.

The enhanced gradient helps overcome the problem of having hidden neurons learning near-identical features. It was able to speed up the overall learning significantly. Also, unlike the traditional gradient rules which were dependent on the representation of the data samples, the enhanced gradient which was derived to be invariant to the representation could successfully learn very dense data set without any difficulty.

The chapter mainly focused on parallel tempering learning, but showed that contrastive divergence learning is also improved by adopting the proposed improvements.

Although the theoretical background suggests that the robust learning rate is well-suited for learning BMs, the experiments were only done with RBMs and a limited number of data sets. It is expected that the proposed learning rule will improve and ease training more generalized BMs such as DBMs and fully-connected BMs.

The application of the proposed adaptive learning rate and the enhanced gradient in Gaussian-Bernoulli RBMs will be discussed more in Chapter 5.

Chapter 5

Restricted Boltzmann Machines for Continuous Data

The conventional RBM assumed that the state of each neuron is binary, e.g. $\{0, 1\}$. It seriously limits the application area of RBMs, as many available data are real-valued. Although there have been attempts to use the binary RBM to learn the real-valued data set by scaling the values to $[0, 1]$ and considering each value as a probability (Hinton & Salakhutdinov, 2006), it has not been used widely.

There have been two notable approaches to overcome this limitation, and they both replace the binary visible neurons with neurons that follow other types of distributions. One approach adopts Gaussian visible neurons and proposes a Gaussian-Bernoulli Restricted Boltzmann Machine (GBRBM). The other approach replaces the binary visible neuron with the softmax unit (Salakhutdinov, 2009a). The former is more appropriate for real-valued continuous data, and the latter for the discrete data with the small number of possible states.

This chapter mainly focuses on GBRBMs and proposes a novel modification to them in order to improve learning. The modification is applied to the energy function of the model and to the gradient learning rules.

5.1 Gaussian-Bernoulli RBM

Let $\mathbf{v} = [v_i]$ be real-valued Gaussian neurons with biases $\mathbf{b} = [b_i]$ and variances $\boldsymbol{\sigma} = [\sigma_i]$. Identical to the conventional binary RBM, let us assume that each hidden neuron h_i can be either 0 or 1 and that it has a bias c_i. Then, the energy of GBRBM given the model and the values for the visible and hidden neurons is traditionally defined as (see e.g. Krizhevsky,

2009; Salakhutdinov, 2009a)

$$E(\mathbf{v}, \mathbf{h}|\theta) = \sum_{i=1}^{n_v} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} W_{ij} h_j \frac{v_i}{\sigma_i} - \sum_{j=1}^{n_h} c_j h_j. \tag{5.1}$$

Then, the conditional probabilities for each visible and hidden neuron given the others are derived to be

$$p(v_i = v|\mathbf{h}) = \mathcal{N}\left(v|b_i + \sigma_i \sum_j h_j W_{ij}, \sigma_i^2\right), \tag{5.2}$$

and

$$p(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(c_j + \sum_i W_{ij} \frac{v_i}{\sigma_i}\right), \tag{5.3}$$

where $\mathcal{N}(\cdot \mid \mu, \sigma^2)$ denotes the pdf of the Gaussian distribution with mean μ and variance σ^2, and $\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$.

By taking the partial derivative of the log-likelihood function which can be derived by marginalizing out the hidden neurons from the joint probability density of the model with respect to each parameter, it is possible to obtain the gradient update rule. Obtaining the update rules is similar to that for the original binary RBM and is omitted in this chapter (see Krizhevsky (2009) for details).

The update rules for the weight w_{ij}, the visible bias b_i, the hidden bias c_j, and the standard deviation σ_i are, then,

$$\nabla W_{ij} = \frac{1}{\sigma_i}\left(\langle v_i h_j \rangle_{\text{d}} - \langle v_i h_j \rangle_{\text{m}}\right), \tag{5.4}$$

$$\nabla b_i = \frac{1}{\sigma_i^2}\left(\langle v_i \rangle_{\text{d}} - \langle v_i \rangle_{\text{m}}\right), \tag{5.5}$$

$$\nabla c_j = \langle h_j \rangle_{\text{d}} - \langle h_j \rangle_{\text{m}}, \tag{5.6}$$

$$\nabla \sigma_i = \frac{1}{\sigma_i^3}\left(\left\langle (v_i - b_i)^2 - \sigma_i \sum_j v_i h_j w_{ij} \right\rangle_{\text{d}} - \left\langle (v_i - b_i)^2 - \sigma_i \sum_j v_i h_j w_{ij} \right\rangle_{\text{m}}\right), \tag{5.7}$$

where $\langle \cdot \rangle_p$ represents the expected value over the distribution p, and d and m are the empirical distribution and the model distribution, respectively.

5.1.1 Practical considerations

GBRBM , in general, is known to be difficult to train. This difficulty mainly comes from learning standard deviations σ_i of the visible neurons.

Unlike other parameters such as w_{ij}, b_i, and c_j, the standard deviations are constrained to be positive. However, with an inappropriate learning rate, it is possible for the obtained gradient update rule to result in a non-positive standard deviation. This leads either to the infinite energy of the model (in case of $\sigma_i = 0$) or to the ill-defined conditional distribution of the visible neuron (in case of $\sigma_i < 0$).

Since all gradients other than that of the hidden bias are scaled by the standard deviation, inappropriate learning of it affects learning of other parameters, also. Too rapid decrease of the standard deviation increases the gradients of the weights and the visible biases such that the stochastic gradient learning either diverges or converges very slowly.

In order to overcome this problem of learning the standard deviations Krizhevsky (2009) suggested using a separate learning rate for the standard deviations which should be 100 to 1000 times smaller than that of the other parameters. However, this imposes a problem of how to choose the right learning rate.

There has been a general consensus that it is enough to update the weights and the biases only, and use fixed, possibly unit, standard deviations. Many impressive results using GBRBMs without learning standard deviations have been published recently (Salakhutdinov, 2009b; Mohamed & Hinton, 2010; Krizhevsky, 2009).

Furthermore, instead of sampling from the Gaussian distribution of the visible neurons, the mean vectors are commonly used as the samples from the visible neuron. It is due to the fact that the standard deviations are not updated, and thus, the samples of the visible neurons are either dominated by the noise or affected only marginally by the standard deviations.

The strategy of no sampling for the visible neurons is possible, since the stochastic nature of the Boltzmann machine could be maintained by the stochastic hidden neurons only rather than by both visible and hidden neurons. It could also be thought analogue to the mean-field approximation for the binary RBM (Welling & Hinton, 2002).

5.2 Improved Learning of Gaussian-Bernoulli RBM

5.2.1 Modified Energy Function

The traditional energy function of GBRBM in Equation (5.1) introduced an unintuitive conditional distribution of visible neurons described in Equations (5.2). The problem comes from the fact that the noise level defined by σ_i affects the mean of the visible neuron, and thus, the noise level cannot be considered solely as a noise, but also as a scaling, or importance, factor for the weights. The role of σ_i as an importance factor can be observed from the conditional distribution of the hidden neurons such that when σ_i is large, the contribution of the value of the visible neuron is small, and vice versa.

In other words, σ_i in the first term of Equation (5.1) denotes a noise level of v_i given the hidden neurons, but the same σ_i in the second term acts as an importance factor for the i-th visible neuron.

Furthermore, the update rules for weights and visible biases in Equations (5.4)–(5.5) are scaled by σ_i, but with different exponents. This could potentially affect the gradient update as σ_i decreases.

Therefore, a modified energy function of GBRBM that addresses the above mentioned problems is newly defined as:

$$E(\mathbf{v}, \mathbf{h}|\theta) = \sum_{i=1}^{n_v} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} W_{ij} h_j \frac{v_i}{\sigma_i^2} - \sum_{j=1}^{n_h} c_j h_j. \tag{5.8}$$

The only difference can be found in the exponent of σ_i in the second term of the energy function. Instead of using a plain σ_i, the modified energy function uses a squared σ_i.

Under the modified energy function, the conditional probabilities for each visible and hidden neurons given others are

$$p(v_i = v|\mathbf{h}) = \mathcal{N}\left(v|b_i + \sum_j h_j W_{ij}, \sigma_i^2\right), \tag{5.9}$$

and

$$p(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(c_j + \sum_i W_{ij}\frac{v_i}{\sigma_i^2}\right). \tag{5.10}$$

Comparing these to (5.2) – (5.3), the obvious change can be observed from the conditional distribution of visible neuron v_i, where the standard deviation σ_i does not appear in the

mean of the Gaussian distribution. The conditional distribution of hidden neuron h_j was modified such that the value of each visible neuron is now scaled by the square of σ_i.

The update rules for the parameters are, then,

$$\nabla W_{ij} = \frac{1}{\sigma_i^2}\left(\langle v_i h_j\rangle_{\mathrm{d}} - \langle v_i h_j\rangle_{\mathrm{m}}\right), \tag{5.11}$$

$$\nabla b_i = \frac{1}{\sigma_i^2}\left(\langle v_i\rangle_{\mathrm{d}} - \langle v_i\rangle_{\mathrm{m}}\right), \tag{5.12}$$

$$\nabla c_j = \langle h_j\rangle_{\mathrm{d}} - \langle h_j\rangle_{\mathrm{m}}, \tag{5.13}$$

$$\nabla \sigma_i = \frac{1}{\sigma_i^3}\left(\left\langle (v_i - b_i)^2 - 2\sum_j v_i h_j w_{ij}\right\rangle_{\mathrm{d}} - \left\langle (v_i - b_i)^2 - 2\sum_j v_i h_j w_{ij}\right\rangle_{\mathrm{m}}\right), \tag{5.14}$$

where all the symbols follow the convention of those used for defining the update rules based on the traditional energy function.

It is clear to see that the gradients for both weights and visible biases are now scaled identically by σ_i^{-2}. Also, all terms in the gradient of σ_i are now scaled equally by σ_i^{-3}, whereas the gradient obtained from the conventional energy function two terms were scaled with the inverse of a square, and the others were with the inverse of a cube.

Learning log-standard deviation

In addition to the proposed modified energy function, this section proposes to re-parameterize the variance σ_i^2 with an exponentiated new variable z_i in the energy function of GBRBM such that

$$E(\mathbf{v}, \mathbf{h}|\theta) = \sum_{i=1}^{n_v} \frac{(v_i - b_i)^2}{2e^{z_i}} - \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} W_{ij} h_j \frac{v_i}{e^{z_i}} - \sum_{j=1}^{n_h} c_j h_j.$$

Then, the square of standard deviation of the conditional distribution of each visible neuron v_i is e^{z_i}, and $z_i = \log \sigma_i^2$. The gradient of z_i is

$$\nabla z_i = e^{-z_i}\left(\left\langle \frac{1}{2}(v_i - b_i)^2 - \sum_j v_i h_j w_{ij}\right\rangle_{\mathrm{d}} - \left\langle \frac{1}{2}(v_i - b_i)^2 - \sum_j v_i h_j w_{ij}\right\rangle_{\mathrm{m}}\right).$$

The gradient is now scaled identically to the other parameters by σ_i^{-2}

As discussed earlier in this chapter, there have been papers proposing not to update σ_i when training GBRBMs. Krizhevsky (2009) reported that learning standard deviations of visible neurons tremendously decreased the reconstruction error[1] while also claimed that it made filters learned by GBRBM more noisy. Also, Salakhutdinov (2009a) claimed that σ_i would be fixed to a predetermined value in practice. It is, however, not easy to have a universal method for determining the appropriate values.

In the experiments that will be presented later in this chapter the effect of learning standard deviations will be discussed.

5.2.2 Parallel Tempering

Although CD learning can be applied directly to GBRBMs without any modification other than the gradient update rules, parallel tempering learning needs to be redefined.

The main problem is that the usual practice of multiplying each parameter with the temperature, as described in Chapter 3, does not apply in the case of GBRBM , because of the standard deviations σ_i.

A naive approach of multiplying σ_i with the temperature results in the base model (which is the model with the highest temperature 0) having visible neurons that have zero standard deviations. On the other hand, if tempering is considered to be done on the energy function such that the energy of GBRBM with its temperature t is defined as $tE(\mathbf{v}, \mathbf{h})$, the standard deviation of each visible neuron at the base model approaches infinity.

In order to overcome this problem, a new scheme for constructing the intermediate tempered GBRBM is proposed such that its parameters are given by

$$
\begin{aligned}
W_{ij}^{(t)} &= tW_{ij}, \\
b_i^{(t)} &= tb_i + (1-t)m_i, \\
c_j^{(t)} &= tc_j, \\
\sigma_i^{(t)} &= \sqrt{t\sigma_i^2 + (1-t)s_i^2},
\end{aligned}
$$

where W_{ij}, b_i and c_j are the parameters of the model, and the superscript (t) indicates that they are of the tempered intermediate model with the temperature t, respectively.

m_i and s_i^2 are the mean and the variance of the i-th component of the training data samples,

[1] Refer to Chapter 2 for its definition.

and they are defined by

$$m_i = \frac{1}{N}\sum_{n=1}^{N} x_i^{(n)}, \qquad s_i = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}\left(x_i^{(n)} - m_i\right)^2},$$

in which $x_i^{(n)}$ is the i-th component of the n-th training sample in the training data set $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$.

In this proposed scheme, the intermediate model is the result of interpolating the base model and the current model, and the base model consists of independent Gaussian variables that has the means and variances of the training data samples.

5.2.3 Adaptive Learning Rate

As it was pointed out in Chapter 4, training RBMs is often sensitive to the choice of learning rate and its scheduling. According to the experiments of which the results will be shown later in this chapter, GBRBM tends to be more sensitive to the choice than RBM is. Also, various experiments showed that if the learning rate is not annealed over time approaching zero, GBRBMs diverges easily after initial convergence.

This problem can be, in practice, easily addressed by limiting the maximum learning rate. In all the experiments in the rest of this chapter, the adaptive learning rate is limited from above such that the learning rate is taken to be $\max(\eta, \bar{\eta})$.

Additionally to limiting the learning rate from above, a lower-bound can also be set. This possibly prevents the adaptive learning rate from getting stuck in the area where the sufficiently small neighborhood does not improve the local likelihood estimate[2].

5.2.4 Enhanced Gradients

The enhanced gradient was proposed in Chapter 4 for improving both the speed and the resulting performance, in terms of log-likelihood and feature extracting capability, for the conventional RBM. The same enhanced gradient can be applied to GBRBMs for enhancing learning, however, with a slight difference in deriving the enhanced rules.

[2]This strategy can directly be applied to the conventional binary RBM as well, although the experiments presented in Chapter 4 showed that PT learning with the swapping interval n_{swap} set to 1 prevents the divergence of the learning rate, which minimizes the need for this strategy when training binary RBMs.

Unlike RBM the visible neuron of GBRBM is not binary, but real-valued. It is not possible to apply the bit-flipping transformation to the visible neuron, but only to the hidden neuron. Hence, a different transformation to which the traditional update rules for GBRBMs are not invariant can be defined as follows:

$$\tilde{v_i} = v_i - \mu_i,$$

where $\tilde{v_i}$ is a shifted version of original v_i.

Identically to how the gradient update rule for the weights of RBM was rewritten, the gradient update rule for the weights of GBRBM can also be rewritten as:

$$\nabla w_{ij} = \mathrm{Cov_d}\left(v_i, h_j\right) - \mathrm{Cov_m}\left(v_i, h_j\right) + \left\langle v_i \right\rangle_{\mathrm{dm}} \nabla c_j + \left\langle h_j \right\rangle_{\mathrm{dm}} \nabla b_i \,,$$

where ∇c_j and ∇b_i are the gradients defined in (5.5)–(5.6) and $\left\langle \cdot \right\rangle_{\mathrm{dm}} = \frac{1}{2}\left\langle \cdot \right\rangle_{\mathrm{d}} + \frac{1}{2}\left\langle \cdot \right\rangle_{\mathrm{m}}$ is the average activity of neuron under the data and model distributions. Remarkably, it is identical to that of the conventional RBM.

In a similar manner, the enhanced gradient rules are obtained by transforming the energy, updating the transformed energy, and transforming it back. They, then, remain in the same forms as they were for RBM with both binary visible and hidden neurons.

In the case of standard deviations σ_i, the enhanced gradient obtained from the shifting transformation does not change the update rule. The same update rule in (5.7) can be used.

5.3 Learning Human Faces

In all experiments, the following settings were used. Weights were initialized to uniform random values between $\pm\frac{1}{n_v+n_h}$. Biases b_i and c_j were initialized to zero and variances σ_i to ones. Adaptive learning rate candidates (see Section 4.1) were $\{0.9\eta, \eta, 1.1\eta\}$, where η is the previous learning rate. In PT learning, there were 21 equally spaced temperatures $t \in \{0, 0.05, \ldots, 1\}$, and in CD learning, a single Gibbs step was taken for each update. For images, each pixel was normalized into $[0, 1]$.

The CBCL data used in the experiment contains 2,429 faces and 4,548 non-faces as training set and 472 faces and 23,573 non-faces as test set (MIT Center For Biological and Computation Learning). Since the intention of the experiment is to check whether GBRBM can learn meaningful features of a probabilistic distribution, only the faces from the training set of the CBCL data were used. A set of randomly chosen samples of faces are shown in

Figure 5.1.

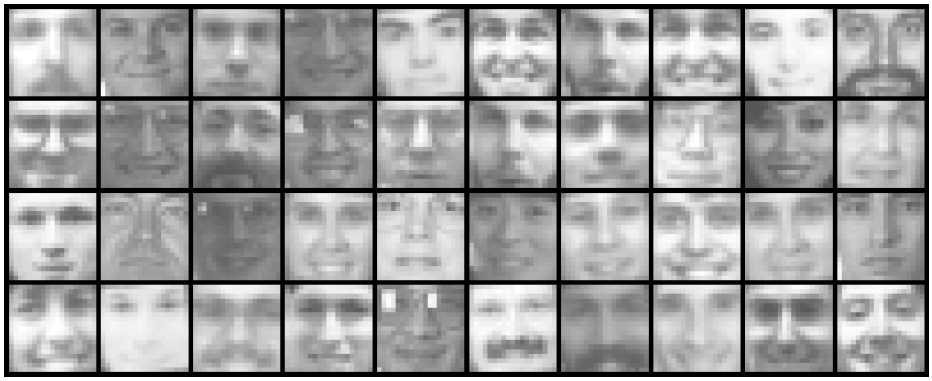
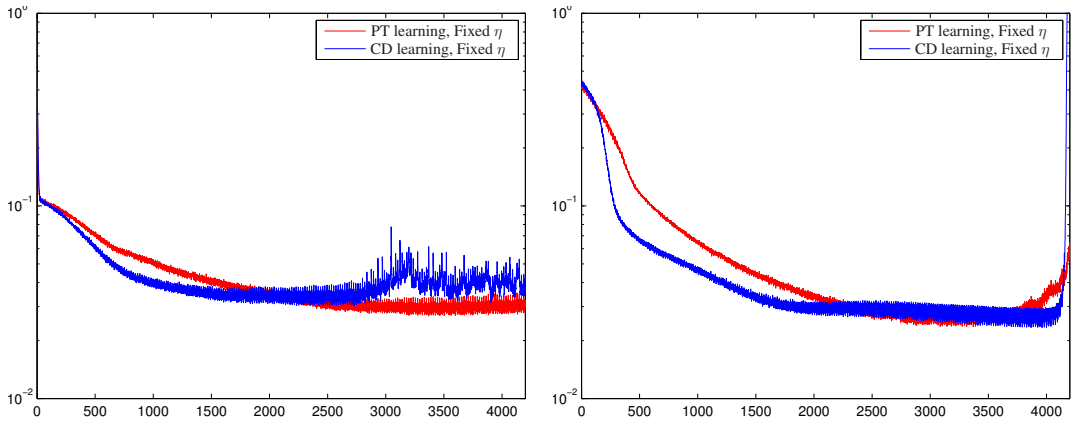


Figure 5.1: 40 randomly chosen faces from CBCL data.

5.3.1 Sensitivity to learning rate scheduling

The aim of the first experiment was to test whether the learning rate scheduling is important and whether GBRBM is more sensitive to it than the conventional binary RBM as discussed in Section 5.2.3.

The procedure was simply to train a GBRBM with 256 hidden neurons using both the traditional gradient and the enhanced gradient with the learning rate fixed to 0.001 while updating both standard deviations and other parameters. The divergence of the GBRBM was observed by monitoring the reconstruction error.



(a) Traditional gradient (b) Enhanced gradient

Figure 5.2: Reconstruction errors obtained by training GBRBMs without using any learning rate scheduling, but a learning rate fixed to 0.001.

As can be seen from Figure 5.2 which shows the reconstruction errors (see Section 2.3.2

for the definition) computed during learning, regardless of the gradients (either traditional or enhanced) or the learning methods (either PT learning or CD learning), the learning diverged after some updates. Out of the expectation, the traditional gradient seems to be more resilient to the divergence, but the mean and the variance of the reconstruction error increased over time, regardlessly. It became more evident after more than 6000 updates, which is not shown in the figure.

This divergence is an evidence for GBRBM 's sensitivity to the learning rate scheduling. Considering that the divergence became significant when the standard deviations decreased significantly (this is not shown in the figures, though) the divergence can be explained by the scaling factors embedded in the gradient update rules which are highly dependent and increase exponentially with respect to the values of the standard deviations.

Further experiments were, thus, performed with the adaptive learning rate in order to automatically anneal the learning rate. As will be shown, when the learning rate is annealed, the divergence was not observed, emphasizing the importance of using the adaptive learning rate.

5.3.2 Learning standard deviation is important

The trained GBRBM had 256 hidden neurons. Initially, the standard deviations of the visible neurons were not updated, but fixed to the constant value of 1. The training was performed for approximately 650 epochs which is equivalent to around 12466 gradient updates. The training was performed by CD learning with a single Gibbs sampling step to obtain the model samples.

The enhanced gradient and the adaptive learning rate were used, and the initial learning rate was 0.0001, and the upper-bound and the lower-bound were set to 0.01, and 0.0001, respectively.

Figure 5.3 shows the learned filters and samples generated from the GBRBM after approximately 650 epochs. The reconstruction error nearly converged (see Figure 5.5), but it is clear that the samples are very noisy. Additionally to the samples, the filters learned by the GBRBM are quite noisy, possibly suggesting that there is more room for training further.

Further training continued, however, now with updating the standard deviations of the visible neurons for 1000 epochs. From both the weights and the generated samples, it is obvious that the noise presented previously has been reduced tremendously.

From the reconstruction error shown in the left figure of Figure 5.5, it is clear that learn-

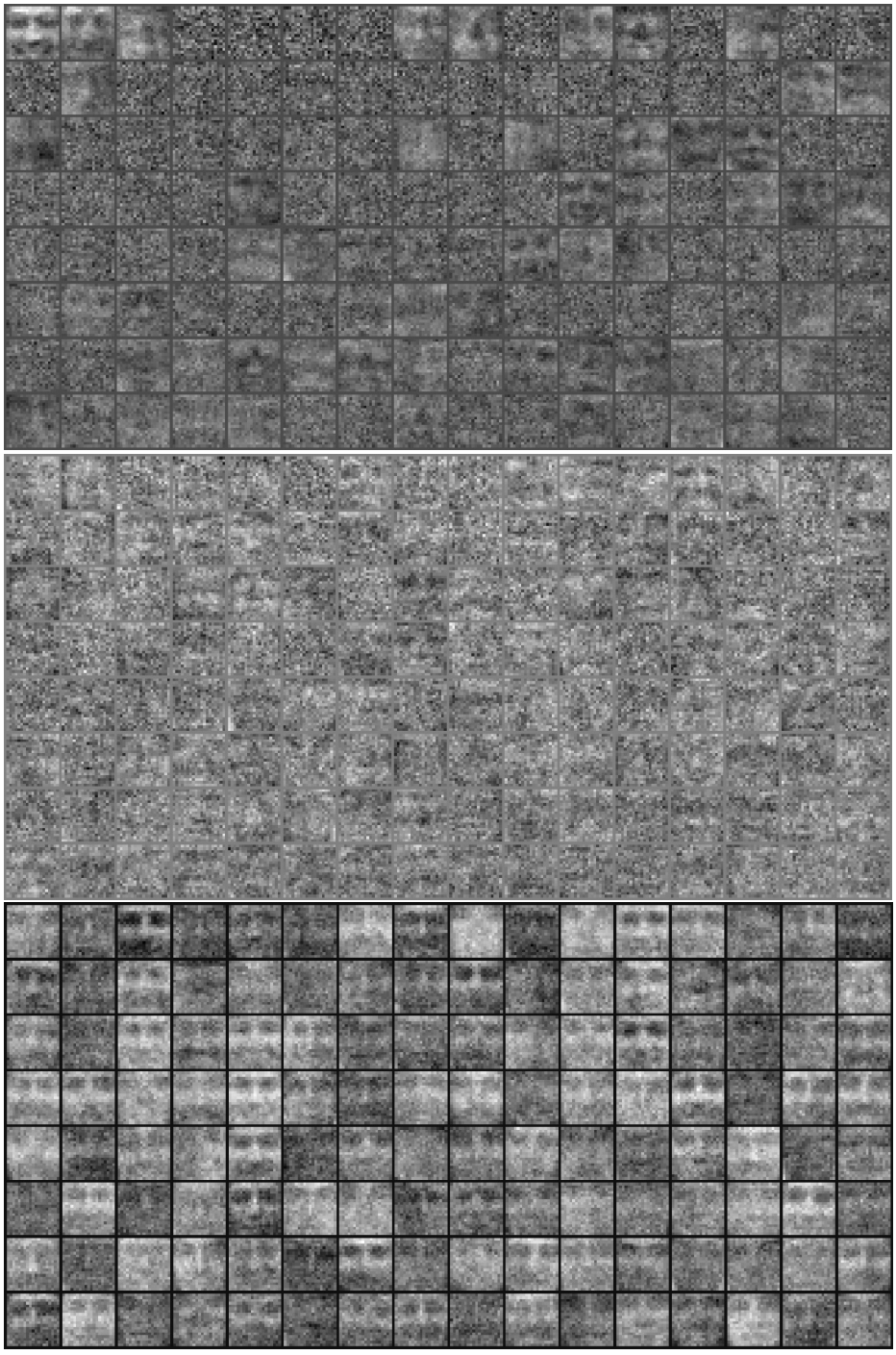


Figure 5.3: Filters (top, middle) and samples (bottom) generated by the GBRBM trained without updating standard deviations. The filters were sorted with respect to the L2-norms such that 128 filters with the large norms are shown in the top figure and the others are in the middle figure (from top to bottom, left to right). Between each consecutive samples 100 Gibbs sampling steps were performed.

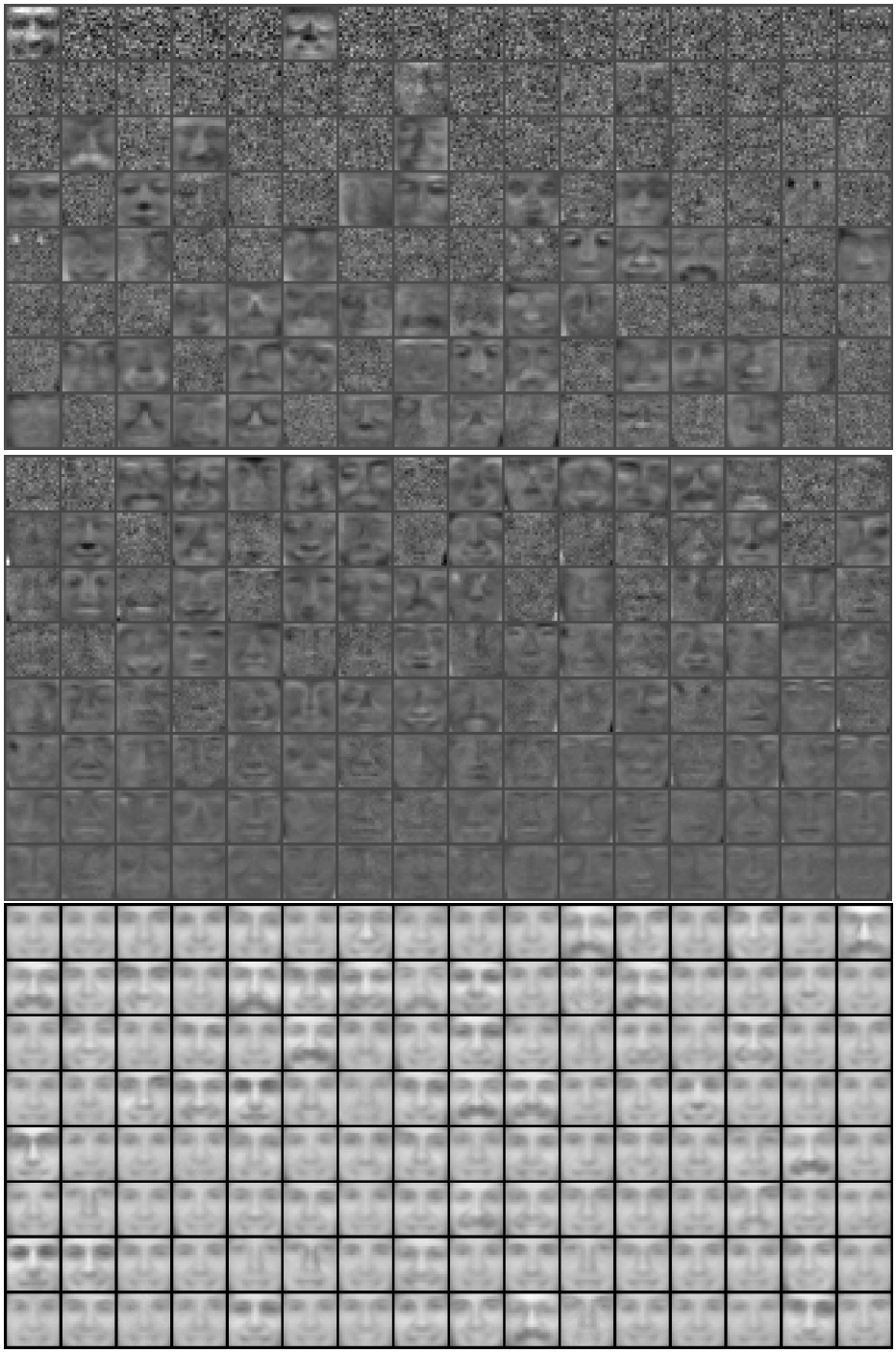


Figure 5.4: Filters (top) and samples (bottom) generated by the GBRBM that was continued to be trained now with updating the standard deviations. The filters were sorted with respect to the L2-nors. Between each consecutive samples 100 Gibbs sampling steps were performed.

ing the standard deviations decreases the reconstruction error immediately. The underlying reason for this behavior could be explained as the GBRBM has become aware of the importance among visible neurons so that it emphasizes those pixels that are more important while modeling the training data set.

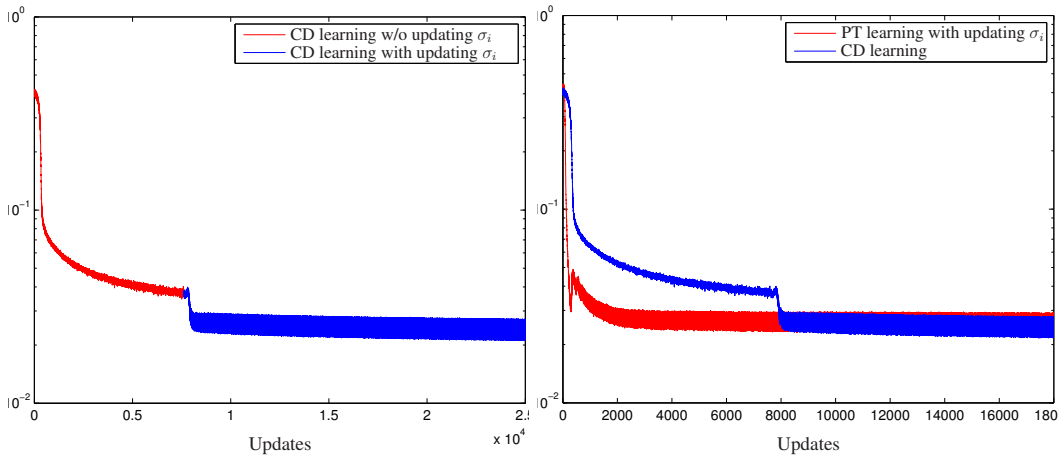


Figure 5.5: Evolutions of reconstruction error over gradient updates.

It is interesting to see the visualization of the learned standard deviations which is shown in Figure 5.6. It is interesting to see that those parts of a face that are important for the recognition such as eyes and a mouth have lower standard deviations while other parts such as both chins have higher standard deviations that are close to the standard deviations of the training samples. It corresponds to the earlier explanation of σ_i being an importance factor as GBRBM focuses more on those important parts when modeling the faces, and since those parts are rather well modeled, given the values of the hidden neurons, the noise levels of visible neurons are significantly lower.

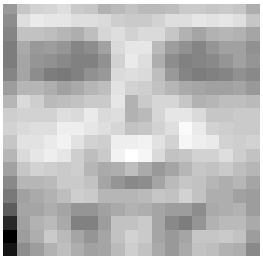


Figure 5.6: Learned standard deviations after approximately 1645 epochs using CD learning.

5.3.3 Parallel tempering for training Gaussian-Bernoulli RBM

In order to see (1) if there is any risk in learning standard deviations from the beginning and (2) if the use of PT learning with the intermediate distributions proposed in Section 5.2.2 works, an additional experiment was conducted. A GBRBM with the same number of hidden neurons was trained using PT learning while updating the standard deviations from the very first gradient update.

The observation of the reconstruction error on the right-hand figure of Figure 5.5 suggests that learning the standard deviations from the beginning indeed helps. Also, it is apparent that the learning does not diverge thanks to the adaptive learning rate.

In addition to the reconstruction error that revealed that PT learning with the proposed intermediate distribution works well, the samples were generated from the trained GBRBM. Visual inspection of the generated samples in Figure 5.7 and Figure 5.4 suggests that the GBRBM trained using PT learning is more suitable for generating a richer variety of samples, which indirectly indicates that a better generative model was learned by PT learning.

5.4 Learning Features from Natural Image Patches

An experiment was conducted in order to see if the learned GBRBM can be used as a feature extractor.

CIFAR-10 data set (Krizhevsky, 2009) which consists of three-channel (R, G, B) color images of size 32×32 with ten different labels[3] was divided into three sets which are training, validation, and test data sets. They contain 40000, 10000, and 10000 images, respectively. Some of the sample images are shown in Figure 5.8.

5.4.1 Learning image patches with CD and PT learning

In this experiment, the procedure proposed by Ranzato & Hinton (2010) is roughly followed which was successfully used for classification tasks (Krizhevsky, 2009, 2010; Coates et al., 2010). The procedure, first, trains GBRBM , or any other feature extractor of the choice, on small image patches (see Figure 5.9 for a number of examples).

Two GBRBMs, each having 300 hidden neurons, constructed under the modified energy function were trained on 8×8 images patches of which each pixel consists of 3 real values

[3]Labels are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

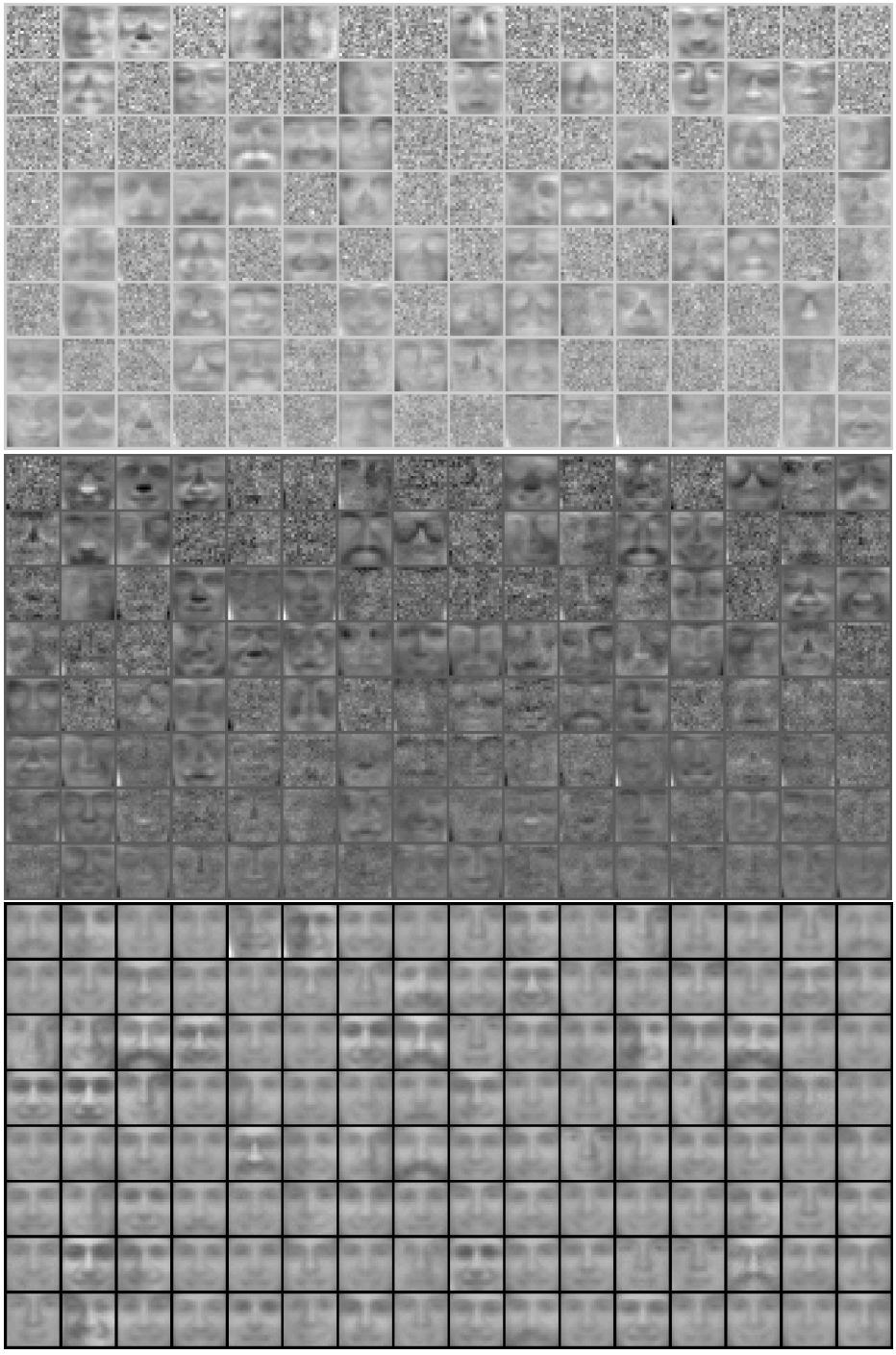


Figure 5.7: Filters (top) and samples (bottom) generated by GBRBM trained without updating standard deviations. The filters were sorted with respect to the L2-norms. Between each consecutive samples 100 Gibbs sampling steps were performed.

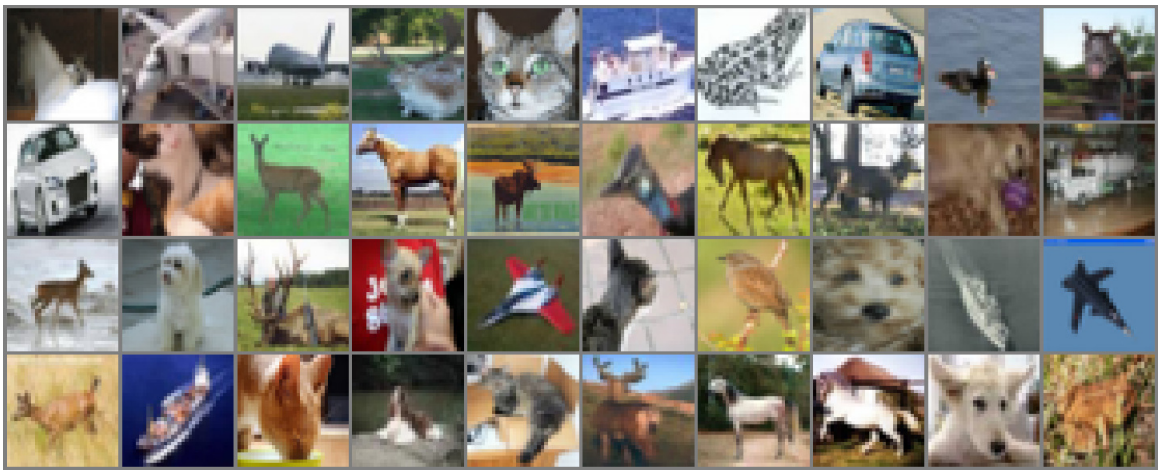


Figure 5.8: 40 randomly chosen samples from CIFAR-10 data set.

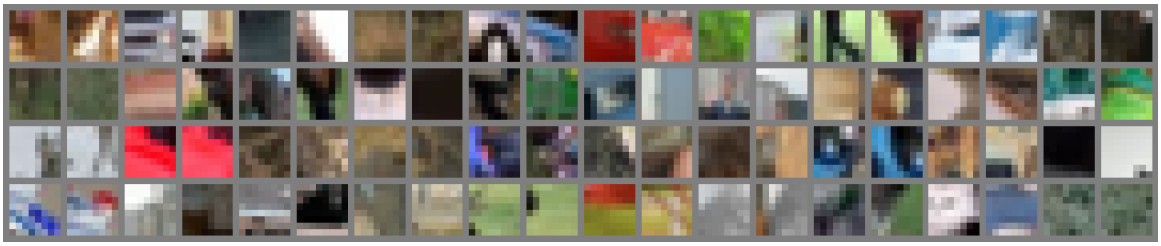


Figure 5.9: 80 randomly chosen samples out of image patches extracted from CIFAR-10.

corresponding to red, green, and blue color components. One GBRBM learned the patches using CD learning, and the other one did using PT learning. The proposed enhanced gradient and the adaptive learning rate were used by both GBRBMs.

The GBRBM trained using PT learning was updated for 200 epochs, and the other was updated for 300 epochs. No preprocessing was performed for the patches other than normalizing each color component into $[0, 1]$.

Figures 5.10–5.11 visualize the filters learned by the GBRBMs. It is clear that the filters with the large norms learn mostly the global structure of the image patches, whereas those with the smaller norms tend to model more details, regardless of the learning method.

It is notable that the GBRBM was able to learn both the straight edge filters and the curved edge filters. It is more obvious in case of PT learning, whereas in case of CD learning, the filters with the small norms mostly learned not-so-useful global structures.

Furthermore, the GBRBM favored high frequency edge-like filters for black-and-white filters, whereas on the other hand, the low frequency filters that model more global features show the variety of colors. This can be explained so that the image patches can be modeled by the combination of the global color patterns and the position information of edges

Figure 5.10: (Top) 128 filters with the largest L2-norms, (middle) 128 filters with the least L2-norms, and (bottom) 90 samples where each consecutive samples are separated by 100 Gibbs sampling steps, obtained by training a GBRBM on natural image patches using PT learning.

(Krizhevsky, 2009).

The standard deviations were distributed in $[0.1681\ 0.2074]$ and $[0.1756\ 0.1932]$ for GBRBMs trained with PT learning and CD learning, respectively. In both cases, the learned standard deviations were significantly smaller than those of the training samples, which were distributed between 0.2338 and 0.2641. This was expected and is desirable.

5.4.2 Learning features for classifying natural images

For the actual classification task, 49 patches were obtained in a convolutional way for each image. Each patch was, then, preprocessed and converted to 64 independent components by the already obtained independent component analysis (ICA) filters. The activation probabilities of the 200 or 300 hidden neurons of GBRBMs were obtained with the 64 components and used for the classification. The classification was done by the simple logistic regression.

To classify the color image, independent component analysis (ICA) (Hyvärinen et al., 2001) and GBRBM were trained on the randomly chosen patches from the training data set. More

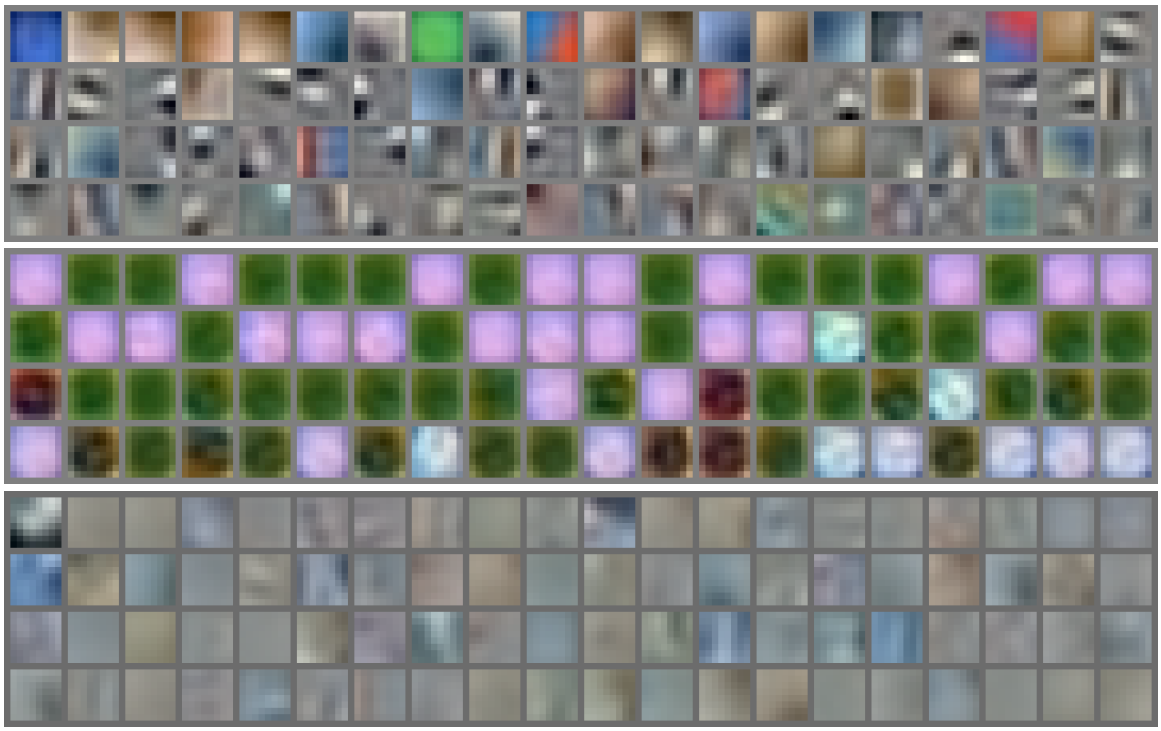


Figure 5.11: (Top) 128 filters with the largest L2-norms, (middle) 128 filters with the least L2-norms, and (bottom) 90 samples where each consecutive samples are separated by 100 Gibbs sampling steps, obtained by training GBRBM on natural image patches using CD learning.

precisely three patches were randomly chosen from each training image to obtain the mixing and separating matrices of ICA where the hyperbolic tangent function was used as the non-linearity function and the components were estimated symmetrically (see Figure 5.12 for the obtained separating matrix).

The algorithm for ICA was FastICA (Hyvärinen, 1999). For the efficient training and the removal of any possible noise, only the first 64 components of the largest eigenvalues were retained at the whitening phase of the ICA training, and therefore, 64 independent components were obtained after ICA.

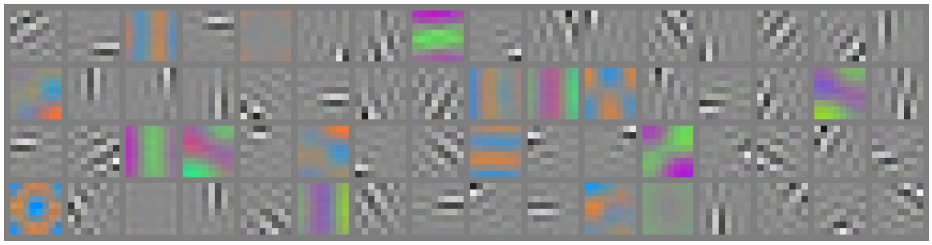
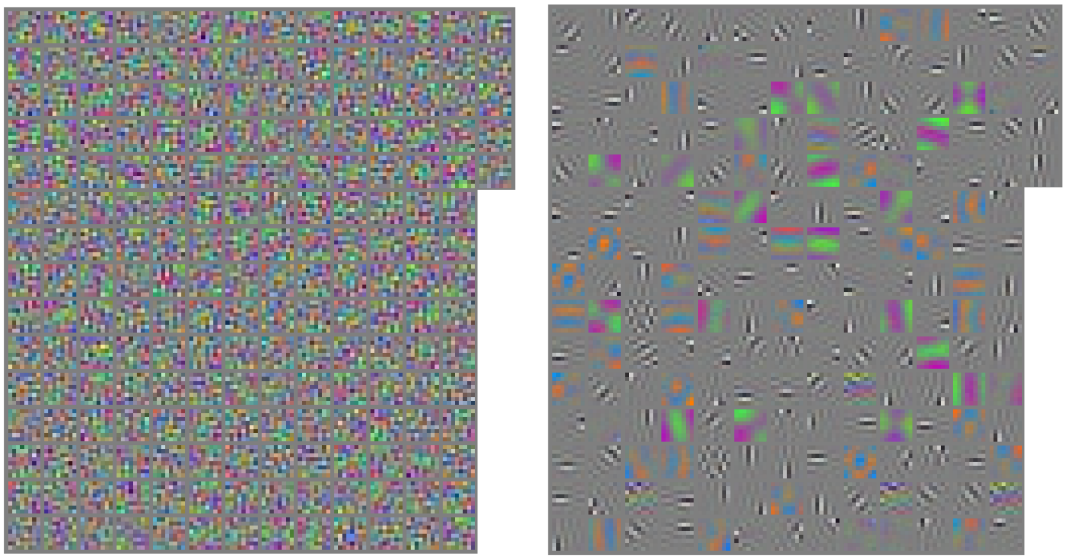


Figure 5.12: 64 filters (separating matrix) obtained by ICA on 8×8-patches of the natural images.



(a) GBRBM filters - before learning (b) GBRBM filters - after learning

Figure 5.13: (Left) filters of ICA+GBRBM before training, and (right) filters by ICA+GBRBM learned from the natural image patches. GBRBM filters were visualized by ICA projection.

The independent components obtained by ICA for each training image were used as the training data for a GBRBM. The GBRBM had 200 or 300 binary hidden neurons, and was trained by PCD learning using the traditional learning rules without updating the standard deviations, but fixing them to 1. The minibatch of size 20 was used.

The traditional gradient rules were used instead of the proposed enhanced gradient rules, since some preliminary experiments suggested that the traditional gradient rules performed without any problem if GBRBM learned the independent components. Further, the learning rate was fixed to 0.005 for the whole training and for all models instead of the adaptive learning rate, as the sensitivity to the learning rate scheduling is mostly influenced by the learning of standard deviations which, in this experiment, are not updated, but fixed to 1.

Figure 5.13 shows 200 filters learned by the GBRBM. It is worthwhile to note that the filters expanded by the GBRBM after ICA shows more variety of the edge-like filters obtained by ICA shown in Figure 5.12. Also, some of the filters by the GBRBM are the combinations of ICA filters.

The best classification accuracy of 63.75% was achieved with ICA+GBRBM having 64 independent components and 300 hidden neurons after training the GBRBM for only about 35 epochs. The similar accuracy were observed by using whitening only instead of ICA. The accuracy obtained using Whitening+GBRBM with 200 hidden neurons was 62.38%. Only difference that could be observed was the not-so-significant slow-down in the convergence

and the marginally worse final accuracy.

Also, worse accuracies could be achieved if the image patches were not preprocessed with ICA nor whitening. Using the filters obtained in the previous experiment (in Section 5.4), the accuracies were 57.42% and 55.20% for PT learning and CD learning, respectively. This suggests that the appropriate preprocessing of the samples is important for extracting the features using GBRBM .

Despite the minor differences, all the accuracies obtained by using the features extracted by GBRBMs were much higher than the classification accuracy obtained by a simple logistic regression classifier on raw pixel values. Only about 40% accuracy could be achieved when the raw pixels were used as features for the classifier. It suggests that GB-RBMs are also capable of extracting features that are more suitable for the classification task.

The obtained best accuracy is comparable to the previous research. Some of the previous results using the variants of RBM without deep neural networks and fine-tuning include 63.78% obtained by GBRBM without any preprocessing, but whitening (Krizhevsky, 2009), 62.8% obtained by the factored 3-way RBM (Ranzato et al., 2010), and 68.2% obtained by the mean and covariance RBM (mcRBM) with the data preprocessed with PCA (Ranzato & Hinton, 2010). However, the result is far from the current state-of-the-art accuracies, e.g. 79.6% obtained by Coates et al. (2010), or 78.90% obtained by Krizhevsky (2010).

It should be noticed that the mentioned results by other researchers were obtained by using the images other than those contained in CIFAR-10 data set. All of them used a non-overlapping set of images from Tiny Images data set[4] which is the unlabeled superset of CIFAR-10. This kind of including other unlabeled data, which can be regarded as a semi-supervised learning, has been shown to improve the generalization performance of the model as well as the classification performance (Krizhevsky, 2009; Ranzato & Hinton, 2010; Ranzato et al., 2010; Hinton & Salakhutdinov, 2006; Salakhutdinov, 2009b). Thus, the performance of the proposed model has the potential for better accuracy if more unlabeled data were used.

5.4.3 Learning images

Due to the difficulty in training GBRBMs, only data sets with comparably small dimensionality have been used in various papers. For instance, one of the most popular benchmark

[4]http://groups.csail.mit.edu/vision/TinyImages/

data sets has been natural image patches (Krizhevsky, 2009, 2010; Coates et al., 2010; Ranzato et al., 2010; Osindero & Hinton, 2008) which consist of 8×8 images of which each pixel consists of three color channels-red, green, and blue.

In case of CIFAR-10 which was used for experimenting the feature extracting performance of GBRBMs in the previous section, Krizhevsky (2009) asserted that GBRBM was unable to learn any meaningful features from the whole images. Later, Krizhevsky (2010) tried various heuristics to prevent GBRBM from learning filters that are focused on modeling the boundaries of the images.

In this experiment, using the enhanced gradient and the adaptive learning rate while the standard deviations are learned, the whole images of CIFAR-10 are learned by GBRBM . It was expected that the standard deviations which also act as importance factors would prevent GBRBM from focusing too much on *unimportant* boundary pixels, but would encourage it to learn both the boundary and the interior of the images.

A GBRBM with 4000 hidden neurons was trained on the images of CIFAR-10 data set. CD learning with the adaptive learning rate and the enhanced gradient was used.

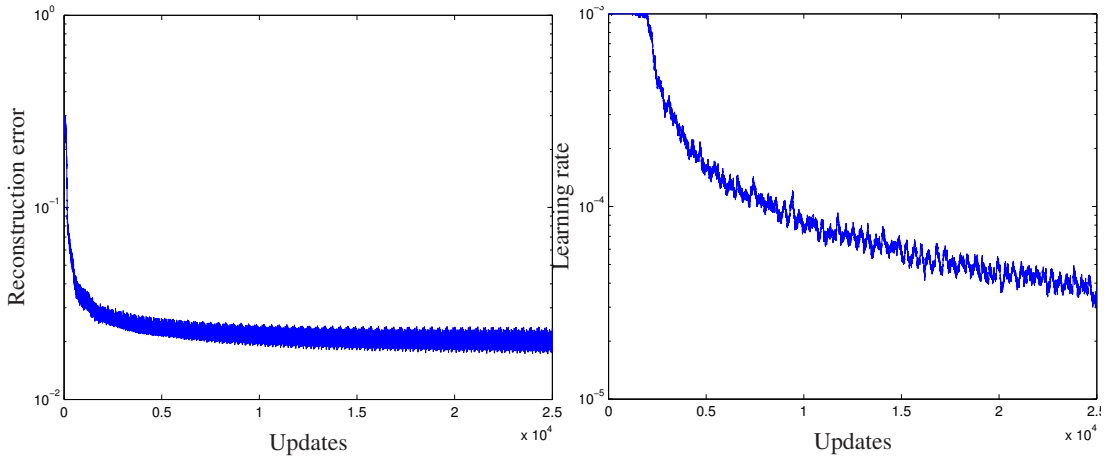
The initial learning rate and the upper-bound were set to 0.001, and no lower-bound for the learning rate was set. The standard deviations were learned from the beginning. The GBRBM was trained for only 70 epochs which is equivalent to 27,370 gradient updates as the minibatch of size 128 were used.

Figure 5.14 show 512 filters learned by the GBRBM sorted by the L2-norms. Clearly, the filters with the large norms tend to model the global features such as the overall background color and the separation between the background seen in the boundary and the object in the middle. Filters with the smaller norms, on the other hand, model those small, fine details mostly concentrated on the interior of the image.

This visualization shows that GBRBM with the modified energy function, the enhanced gradient update, the adaptive learning rate, and learning standard deviations, as proposed in this chapter does not suffer from the problem described by Krizhevsky (2010) which stated that GBRBM easily fails to model the whole image by focusing mostly on the boundary pixels only.

The evolution of the reconstruction error over training shown in the left-hand side of Figure 5.15 shows that regardless of the large dimensionality of the data set which is 3072, the GBRBM was able to learn the images stably. Also, the adaptive learning rate was able to anneal the learning rate appropriately over the training, as can be observed in the right-hand figure of Figure 5.15.

Figure 5.14: Visualization of weights of GBRBM trained on the whole images of CIFAR-10. (Left) 256 filters with the largest L2-norms, and (right) 256 filters with the least L2-norms.



(a) Reconstruction error (b) Learning rate

Figure 5.15: Evolutions of reconstruction error and learning rate while training the GBRBM using CD learning with updating the standard deviations on the whole images of CIFAR-10.

In addition to the visualization of the learned filters and the reconstruction error, it is possible to observe that the GBRBM was able to capture the essence of the training samples by looking at the reconstructed images obtained by a single step Gibbs sampling. Figure 5.16 shows both the randomly chosen original training samples and their reconstructions. The reconstructed images look like blurred versions of the original ones, however, still maintaining the overall structures.

For instance, the reconstructed image of the bottom-left image which has a sedan shows that the GBRBM could capture the uniform background information, the darker color of the bottom of the sedan, and the overall shape of the car. A reconstruction of an eagle flying in the sky (the top third image from the right) captures black wings and white head and tail while ignoring too local, small details. This ignorance can be considered as a sign for either more training being required or more hidden neurons being required.

These results clearly indicate that GBRBM trained with the modified energy function, the enhanced gradient, and the adaptive learning while also learning the standard deviations is able to learn the whole images without much difficulty. Especially, the visualized filters do not possess those filters that can only be considered as global, noisy filters (see Figure 2.1 of Krizhevsky, 2009). Also, clearly most filters do not focus nor model the boundary of the images, which was considered harmful and difficult to address.

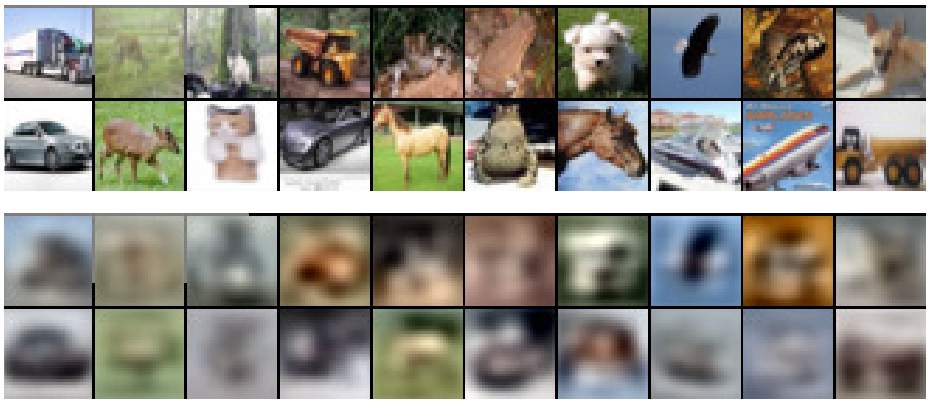


Figure 5.16: (Top) 20 randomly chosen samples from CIFAR-10 data set. (Bottom) One-step reconstruction using the GBRBM.

5.5 Conclusions

This chapter illustrated how RBM can be extended to learning real-valued data by introducing a Gaussian-Bernoulli RBM (GBRBM) which uses Gaussian visible neurons instead of Bernoulli visible neurons of the conventional RBM.

Based on the widely used traditional form of GBRBM, the chapter proposed a modified GBRBM which uses a different parameterization of the energy function. The modification led to more elegant forms for visible and hidden conditional distributions given each other and gradient update rules.

Furthermore, the chapter described how the three advances proposed in Chapters 3 – 4 can be applied to GBRBMs; they are parallel tempering learning, the enhanced gradient, and the adaptive learning rate. To train GBRBMs using the parallel tempering, a method for constructing the intermediate tempered distributions was proposed.

It was shown that the difficulty of preventing the divergence of learning could be addressed by the adaptive learning rate. However, some preliminary experiments (not shown in the thesis) revealed that training GBRBMs using the adaptive learning rate is highly sensitive to the associated learning parameters, and in most cases either the learning rate or the reconstruction error diverged. This problem was addressed by simply having the predetermined upper bound of the learning rate.

The enhanced gradient which was proposed earlier in Chapter 4 was applied to GBRBMs. As a way for adapting it to Gaussian visible neurons, the shifting transformation was proposed. However, it must be reminded that the shifting transformation may not be the optimal one for GBRBM . There certainly exists a room for other transformations that would

give better performance over the shifting transformation. Hence, further investigation is required.

Finally, the use of GBRBM and the proposed modifications were tested through the series of experiments on realistic data sets including human faces and natural images. Those experiments showed that GBRBM is not only possible to learn continuous real-valued data, but similarly to the conventional RBM, it is able to learn interesting features of the data samples so that the obtained features can increase performance in such machine learning tasks as classification.

Despite these successful applications of GBRBM presented in this chapter, training GBRBMs is still more challenging than training RBM. Further research in improving and easing the training will be required.

Chapter 6

Conclusions

Although Boltzmann machines (BM) and restricted Boltzmann machines (RBM) have been introduced already in 1980s, the wide usage of them had to wait until Hinton (2002) introduced contrastive divergence (CD) learning in 2002. The main barrier in the acceptance of RBMs was the difficulty in computing the stochastic gradient for training the model. Thanks to CD learning, the popularity of RBM and its variants grew rapidly, and a whole field called *deep learning* had opened (Bengio, 2009).

Unfortunately, recent papers (see e.g. Schulz et al., 2010; Fischer & Igel, 2010) reported that it is not trival to train a simple RBM as learning can easily diverge. Without careful tuning of learning parameters, even a simple problem of learning handwritten digits fails, which is observed by the decreasing likelihood or the failure of sampling any meaningful digits from the trained model.

In order to address this difficulty in training RBMs, this thesis aimed to provide methods that could ease training from the difficulties and would potentially result in better trained RBMs. The propositions did not concentrate on a single weakness of learning, but considered the solutions from the various angles.

Firstly, the weakness in computing the learning gradient was addressed. PT learning employed an advanced Markov-Chain Monte-Carlo (MCMC) sampling method for replacing the simple Gibbs sampling which has been the sampling method of choice for computing the negative term of the learning gradient. Chapter 3 described how PT learning can be adapted to training RBMs and provided the experimental results showing the superiority of PT learning over the conventional learning method, CD learning.

The second point on which the thesis focused was the problem with the traditional gradient update rules. The close observation into the original update rules revealed that the conven-

tionally used gradients were not invariant to the representation of training samples and also had hidden terms that distract learning. Based on this observation, in Chapter 4 the thesis proposed the enhanced gradient update rules that have the property of invariance to the data representation and remove the distracting terms from the original gradients. Extensive experiments on a realistic data set confirmed that, regardless of the learning method and the training data sets, the enhanced update rules outperformed the traditional ones.

Furthermore, the difficulty of choosing the learning parameters was addressed with the adaptive learning rate in the same chapter. As Fischer & Igel (2010) pointed out, any inappropriate choice of the learning rate results in a diverging behavior. The adaptive learning rate, based on the local estimate of the likelihood, was able to address this problem by automatically adapting the learning rate on-the-fly. It was confirmed with the various experiments.

Lastly, in Chapter 5, the thesis presented how RBMs can be extended to model continuous, real-valued data sets. Based on a Gaussian-Bernoulli RBM (GBRBM) (Hinton & Salakhutdinov, 2006), the chapter proposed modifications and improvements that make GBRBMs readily available to learn high-dimensional data sets easily. The experimental results suggested that GBRBM which in its original form is more sensitive to the learning parameters and is known to be difficult to learn can more easily learn high-dimesional data sets with the proposed improvements. Although the presented results failed to provide any solid numbers that are state-of-the-art, the indirect evidences such as the visualization of the weights, the reconstruction error, and the generated samples, revealed the improvements gained by the proposed methods.

Clearly from the empirical evidences presented throughout the thesis, the proposed improvements for both RBM and its extension GBRBM address the difficulties reported by the researchers. It is expected that the adaptation of these improvements will encourage many other researchers to work on RBMs, and further, on deep learning.

Deep neural networks such as deep belief networks (DBN) (Hinton & Salakhutdinov, 2006), deep Boltzmann machines (DBM) (Salakhutdinov & Hinton, 2009), and convolutional DBN (Lee et al., 2009), have gained popularity, since Hinton & Salakhutdinov (2006) showed that they can be easily trained when each layer of the networks is pretrained, as if it were RBM .

Without pretraining, it is generally considered that learning deep architectures is difficult, if not impossible, except for few exceptional cases such as training an MLP for classification tasks (Ciresan et al., 2010; Martens, 2010). Salakhutdinov & Hinton (2009) even mentioned that MNIST handwritten digits could not be learned successfully by DBMs without

pretraining.

The proposed improvements presented in this thesis mainly focused on training RBMs with a single layer. All experiments were conducted using single layered architectures only without any relaxation of the structural restrictions imposed on RBMs. However, clearly, the theoretical aspects of the three proposed improvements do not restrict their use in more general Boltzmann machines such as DBMs.

It will be an interesting research topic to apply the proposed methods to deep architectures, either as a part of pretraining or as a sole learning method. As the experiments in the thesis have shown the significant improvements and the possibility of much easier training for RBMs, it can be anticipated that the proposed learning methods would help training more generalized and deeper generative architectures.

Additionally, more applications of RBMs need to be discovered and experimented. Including this thesis, most machine learning tasks tackled so far by the deep learning have mainly been the classification tasks of well-known benchmark data sets. Other application areas that could potentially gain improvement from RBM and its variants include missing value reconstruction, collaborative filtering, image segmentation, and clustering of high-dimensional data sets. More research effort will need to focus on diversifying the application areas.

Bibliography

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

Asuncion, A. and Newman, D. J. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Bengio, Y. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, January 2009.

Bengio, Y. and Delalleau, O. Justifying and generalizing contrastive divergence. *Neural Comput.*, 21:1601–1621, June 2009.

Bergstra, J., Frédŕic, B., Turian, J., Pascanu, R., Delalleau, O., Breuleux, O., Lamblin, P., Desjardins, G., Erhan, D., and Bengio, Y. Deep Learning on GPUs with Theano. In *The Learning Workshop*, 2010. Abstract only.

Besag, J. Statistical Analysis of Non-Lattice Data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24(3):179–195, 1975.

Carreira-Perpiñán, M. A. and Hinton, G. E. On Contrastive Divergence Learning. In Cowell, R. G. and Ghahramani, Z. (eds.), *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)*, pp. 3246 – 3253, Barcelona, Spain, July 2010.

Ciresan, D., Meier, U., and Gambardella, L. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *CoRR*, abs/1003.0358, 2010.

Coates, A., Lee, H., and Ng, A. Y. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., and Delalleau, O. Tempered Markov Chain Monte Carlo for training of Restricted Boltzmann Machines. Technical Report 1345, Universitée de Montréal, 2009.

Desjardins, G., Courville, A., and Bengio, Y. Adaptive Parallel Tempering for Stochastic Maximum Likelihood Learning of RBMs. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010a.

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, Sardinia, Italy, 2010b.

Earl, D. J. and Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

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.

Geyer, C. J. Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proc. 23rd Symp. Interface*, pp. 156–163, Seattle, Washington, 1991.

Haykin, S. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 2 edition, July 1998.

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

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

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

Hyvärinen, A. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–34, 1999.

Hyvärinen, A. Some extensions of score matching. *Computational Statistics & Data Analysis*, 51(5):2499–2512, February 2007.

Hyvärinen, A., Karhunen, J., and Oja, E. *Independent Component Analysis*. Wiley-Interscience, May 2001.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto, 2009.

Krizhevsky, A. Convolutional Deep Belief Networks on CIFAR-10. Technical report, Computer Science Department, University of Toronto, 2010.

Le Roux, N. and Bengio, Y. Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, 20:1631–1649, June 2008.

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. Sparse deep belief net model for visual area V2. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 873–880. MIT Press, Cambridge, MA, 2008.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 609–616, New York, NY, USA, 2009. ACM.

Lyu, S. Interpretation and generalization of score matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pp. 359–366, Arlington, Virginia, United States, 2009. AUAI Press.

Mackay, D. J. C. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, June 2002.

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, Sardinia, Italy, 2010.

Martens, J. Deep learning via Hessian-free optimization. In *Proceedings of the 27th Annual International Conference on Machine Learning*, pp. 735–742, Haifa, Israel, 2010.

MIT Center For Biological and Computation Learning. CBCL Face Database #1. URL `http://www.ai.mit.edu/projects/cbcl`.

Mohamed, A. and Hinton, G. E. Phone recognition using Restricted Boltzmann Machines. In *Proceedings of the 35th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4354–4357, Dallas, Texas, 2010.

Müller, A., Schulz, H., and Behnke, S. Topological Features in Locally Connected RBMs. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)*, Barcelona, Spain, July 2010.

Nair, V. and Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th Annual International Conference on Machine Learning*, ICML '10, pp. 807–814, Haifa, Israel, 2010.

Neal, R. Sampling from Multimodal Distributions Using Tempered Transitions. *Statistics and Computing*, 6:353–366, 1994.

Neal, R. M. Annealed Importance Sampling. *Statistics and Computing*, 11:125–139, 1998.

Osindero, S. and Hinton, G. E. Modeling image patches with a directed hierarchy of Markov random fields. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1121–1128. MIT Press, Cambridge, MA, 2008.

Ranzato, M. A. and Hinton, G. E. Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *CVPR*, pp. 2551–2558, 2010.

Ranzato, M., Krizhevsky, A., and Hinton, G. E. Factored 3-Way Restricted Boltzmann Machines for Modeling Natural Images. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

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

Salakhutdinov, R. Learning in markov random fields using tempered transitions. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 22*, 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.

Salakhutdinov, R., Mnih, A., and Hinton, G. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pp. 791–798, New York, NY, USA, 2007. ACM.

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.

Swendsen, R. H. and Wang, J.-S. Replica Monte Carlo Simulation of Spin-Glasses. *Physical Review Letters*, 57(21):2607–2609, Nov 1986.

Teh, Y. W. *Bethe Free energy and contrastive divergence approximations for undirected graphical models*. PhD thesis, University of Toronto, 2003.

Tieleman, T. *Training restricted Boltzmann machines using approximations to the likelihood gradient*. ICML '08. ACM, New York, NY, USA, 2008.

Tieleman, T. and Hinton, G. E. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1033–1040, New York, NY, USA, 2009. ACM.

Welling, M. and Hinton, G. E. A new learning algorithm for mean field boltzmann machines. In *Proceedings of the International Conference on Artificial Neural Networks*, ICANN '02, pp. 351–357, London, UK, UK, 2002. Springer-Verlag.

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

Appendix A

Update Rules for Boltzmann Machines

Let us start from Equation (2.1), however, with $\mathbf{x}$ being split into $\mathbf{v}$ and $\mathbf{h}$ such that

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

Given a training data set $\{\mathbf{v}^{(n)}\}_{n=1}^{N}$, the log-likelihood function (2.3) can be written as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \sum_{n=1}^{N} \log \sum_{\mathbf{h}} P(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta}) \\
&= \sum_{n=1}^{N} \log \frac{\sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})\right\}}{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right\}} \\
&= \sum_{n=1}^{N} \left(\log \sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})\right\} - \log \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right\} \right).
\end{aligned} \tag{A.1}$$

Let θ be one parameter of $\boldsymbol{\theta}$, and then, the update rule for θ can be easily evaluated by taking the partial-derivative of Equation (A.1) with respect to it. Then, the gradient with

respect to θ is

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta} &= \sum_{n=1}^{N} \frac{\partial}{\partial \theta} \left(\log \sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})\right\} - \log \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right\} \right) \\
&= \sum_{n=1}^{N} \left(\frac{\sum_{\mathbf{h}} \frac{\partial\left(-E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})\right)}{\partial \theta} \exp\left\{-E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})\right\}}{\sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})\right\}} - \right. \\
&\qquad \left. \frac{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \frac{\partial(-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}))}{\partial \theta} \exp\left\{-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right\}}{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left\{-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right\}} \right) \\
&= \sum_{n=1}^{N} \left(\left\langle \frac{\partial\left(-E(\mathbf{v}^{(n)}, \mathbf{h} \mid \boldsymbol{\theta})\right)}{\partial \theta} \right\rangle_{P(\mathbf{h} \mid \mathbf{v}^{(n)}, \boldsymbol{\theta})} - \left\langle \frac{\partial\left(-E(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})\right)}{\partial \theta} \right\rangle_{P(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta})} \right). \quad \text{(A.2)}
\end{aligned}$$

Learning is often performed using mini-batches rather than using all training samples at every update. Hence, the terms inside the outermost summation are computed for only a small subset of training samples and are multiplied with an appropriate learning rate.

To obtain different learning rules for each parameter, the negative energy function needs to be differentiated with respect to each parameter. This derivation is universal to both Boltzmann machines and restricted Boltzmann machines, and after the derivations, the update rules (2.4) – (2.6) and (2.9) – (2.11) can be obtained. Similarily, the update rules for Gaussian-Bernoulli RBMs given in (5.4) – (5.7) and (5.11) – (5.14) can be derived.

Appendix B

Enhanced Gradient

B.1 Bit-flipping transformation

By transforming parameters of RBM, the model can practically be made equivalent even when a bit-flipping transformation given in Equation (4.3) is applied. Let us rewrite the energy function of RBM (2.7) when the bit-flipping transformation is applied.

$$\begin{aligned}\tilde{E}(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = &- \sum_i \sum_j v_i^{1-f_i}(1-v_i)^{f_i} h_j^{1-f_j}(1-h_j)^{f_j} w_{ij} \\ &- \sum_i v_i^{1-f_i}(1-v_i)^{f_i} b_i - \sum_j h_j^{1-f_j}(1-h_j)^{f_j} c_j,\end{aligned}$$

where i and j denote indices of the visible and hidden neurons, respectively. Hence, f_i and f_j also represent bit-flipping transformations for the i-th visible neuron and the j-th hidden neuron.

Then, the above energy function can be modified as

$$\begin{aligned}\tilde{E}(\mathbf{v}, \mathbf{h} \mid \boldsymbol{\theta}) = &- \sum_i \sum_j \Big(((1-2f_i)v_i(1-2f_j)h_j w_{ij} + \\ &\qquad f_i(1-2f_j)h_j w_{ij} + f_j(1-2f_i)v_i w_{ij} + f_i f_j w_{ij} \Big) \\ &- \sum_i ((1-2f_i)v_i b_i + f_i b_i) - \sum_j ((1-2f_j)h_j c_j + f_j c_j)\end{aligned}$$

Then, it is possible to gather the terms according to whether the term has both v_i and h_j, only one of them, or none of them. Due to the formulation of the probability function of

RBM in Equation (2.1), any term that does not contain either v_i or h_j can be considered constant and safely ignored. Also, it is possible to rewrite $1-2f_i$ and $1-2f_j$ as $(-1)^{f_i}$ and $(-1)^{f_j}$.

Then, the reformulated energy function looks like

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

From this, it is straightforward to see that Equations (4.4) – (4.6) hold.

B.2 Update Rules based on Bit-flipping Transformation

Let us consider the update of the transformed weights $\tilde{w}_{ij}$. At each update, $\tilde{w}_{ij}$ is updated by $\tilde{w}_{ij}+\eta\nabla\tilde{w}_{ij}$, where η is a learning rate. By the chain rule $\frac{\partial w_{ij}}{\partial\tilde{w}_{ij}}\frac{\partial\left(-\tilde{E}\right)}{\partial w_{ij}}$, the gradient of $\tilde{w}_{ij}$ is, in fact, $(-1)^{f_i+f_j}\left(\left\langle\frac{\partial\left(-\tilde{E}\right)}{\partial w_{ij}}\right\rangle_{\mathrm{d}} - \left\langle\frac{\partial\left(-\tilde{E}\right)}{\partial w_{ij}}\right\rangle_{\mathrm{m}}\right)$.

Hence, the update rule of the weights, when a model is transformed, updated, and transformed back, simply becomes

$$\begin{aligned}w_{ij} &\leftarrow (-1)^{f_i+f_j}\tilde{w}_{ij} + \eta(-1)^{f_i+f_j}\left(\left\langle\frac{\partial\left(-\tilde{E}\right)}{\partial w_{ij}}\right\rangle_{\mathrm{d}} - \left\langle\frac{\partial\left(-\tilde{E}\right)}{\partial w_{ij}}\right\rangle_{\mathrm{m}}\right) \\ &= w_{ij} + \eta(-1)^{f_i+f_j}\left[\left\langle(-1)^{f_i+f_j}v_ih_j + (-1)^{f_i}v_if_j + (-1)^{f_j}h_jf_i\right\rangle_{\mathrm{d}}\right. \\ &\qquad\left. - \left\langle(-1)^{f_i+f_j}v_ih_j + (-1)^{f_i}v_if_j + (-1)^{f_j}h_jf_i\right\rangle_{\mathrm{m}}\right] \\ &= w_{ij} + \eta\left[\langle v_ih_j\rangle_{\mathrm{d}} - \langle v_ih_j\rangle_{\mathrm{m}} - f_i\left(\langle h_j\rangle_{\mathrm{d}} - \langle h_j\rangle_{\mathrm{m}}\right) - f_j\left(\langle v_i\rangle_{\mathrm{d}} - \langle v_i\rangle_{\mathrm{m}}\right)\right], \qquad \text{(B.1)}\end{aligned}$$

which is identical to Equation (4.7) with simple mathematical manipulations.

Now, let us take a look at b_i. It is easy to see that the update rule for b_i is invariant to the transformation, as the update rules for both b_i and $\tilde{b}_i$ are identical to $\langle v_i\rangle_{\mathrm{d}} - \langle v_i\rangle_{\mathrm{m}}$.

Hence, when a model is transformed, updated, and transformed back, each visible bias b_i

becomes

$$
\begin{aligned}
b_i &\leftarrow (-1)^{f_i}(\tilde{b}_i + \eta\nabla\tilde{b}_i) - \sum_j f_j(w_{ij} + \eta\nabla w_{ij}) \\
&= \left((-1)^{f_i}\tilde{b}_i - \sum_j f_j w_{ij}\right) + \left(\eta\nabla b_i - \eta\sum_j \nabla w_{ij}\right) \\
&= b_i + \eta\left[\nabla b_i - \sum_j f_j \nabla w_{ij}\right].
\end{aligned}
$$

It must be reminded that ∇w_{ij} in this context does not refer to the original update rule of the weights (2.4) or (2.9). Rather, it is the additive term in Equation (B.1) which was derived in the same way; transform a model, update a parameter, and transform a model back. Thus, the update rule for b_i is

$$
b_i \leftarrow b_i + \eta\left[\nabla b_i - \sum_j f_j\left(\langle v_i h_j\rangle_{\mathrm{d}} - \langle v_i h_j\rangle_{\mathrm{m}} - f_i\left(\langle h_j\rangle_{\mathrm{d}} - \langle h_j\rangle_{\mathrm{m}}\right) - f_j\left(\langle v_i\rangle_{\mathrm{d}} - \langle v_i\rangle_{\mathrm{m}}\right)\right)\right]
$$

Again, the derived update rule is identical to Equation (4.8). The update rule for c_j can be similarly obtained as that for b_i was obtained, and the derivation is omitted here.

B.3 Obtaining Enhanced Gradients

With the newly derived update rules, it is possible to obtain the enhanced gradients as a weighted sum of the gradients obtained from all possible combinations of bit-flipping transformations. Each gradient is weighted by

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

which is essentially same with Equation (4.10) except for that visible neurons and hidden neurons are separately shown here.

It is apparent that the sum of the weight over all possible combinations of transformations results in 1. An easy example could be constructed by considering the case where only two neurons exist. Then, the weights are computed as shown in Table B.1.

Hence, any term in the gradient that does not depend on (is multiplied by) the transformation f_i or f_j does not change over the weighted sum. Then, the weighted sum of the update rules

f_1	f_2	$\prod_i \langle x_i \rangle_{\text{dm}}^{f_i} \left(1 - \langle x_i \rangle_{\text{dm}}\right)^{1-f_i}$
0	0	$1 - \langle x_1 \rangle_{\text{dm}} - \langle x_2 \rangle_{\text{dm}} + \langle x_1 \rangle_{\text{dm}} \langle x_2 \rangle_{\text{dm}}$
0	1	$\langle x_2 \rangle_{\text{dm}} - \langle x_1 \rangle_{\text{dm}} \langle x_2 \rangle_{\text{dm}}$
1	0	$\langle x_1 \rangle_{\text{dm}} - \langle x_1 \rangle_{\text{dm}} \langle x_2 \rangle_{\text{dm}}$
1	1	$\langle x_1 \rangle_{\text{dm}} \langle x_2 \rangle_{\text{dm}}$
Sum		1

Table B.1: Example of summing the weight

of w_{ij} over all combinations of transformations is

$$
\begin{aligned}
\tilde{\nabla} w_{ij} &= \mathrm{Cov}_{\mathrm{d}}(v_i, h_j) - \mathrm{Cov}_{\mathrm{m}}(v_i, h_j) + \langle v_i \rangle_{\text{dm}} \nabla c_j + \langle h_j \rangle_{\text{dm}} \nabla b_i \\
&\quad - \sum_{f_i, f_j} \langle v_i \rangle_{\text{dm}}^{f_i} \left(1 - \langle v_i \rangle_{\text{dm}}\right)^{1-f_i} \langle h_j \rangle_{\text{dm}}^{f_j} \left(1 - \langle h_j \rangle_{\text{dm}}\right)^{1-f_j} \left(\nabla c_j f_i + \nabla b_i f_j\right) \\
&= \mathrm{Cov}_{\mathrm{d}}(v_i, h_j) - \mathrm{Cov}_{\mathrm{m}}(v_i, h_j) + \langle v_i \rangle_{\text{dm}} \nabla c_j + \langle h_j \rangle_{\text{dm}} \nabla b_i \\
&\quad - \big[\left(1 - \langle v_i \rangle_{\text{dm}}\right) \langle h_j \rangle_{\text{dm}} \nabla b_i + \left(1 - \langle h_j \rangle_{\text{dm}}\right) \langle v_i \rangle_{\text{dm}} \nabla c_j \\
&\qquad + \langle v_i \rangle_{\text{dm}} \langle h_j \rangle_{\text{dm}} \nabla c_j + \langle v_i \rangle_{\text{dm}} \langle h_j \rangle_{\text{dm}} \nabla b_i \big] \\
&= \mathrm{Cov}_{\mathrm{d}}(v_i, h_j) - \mathrm{Cov}_{\mathrm{m}}(v_i, h_j),
\end{aligned}
$$

which is exactly the enhanced gradient presented in Equation (4.11).

It is possible to obtain the enhanced gradients of visible and hidden biases following the identical procedure. However, they can be derived more simply by observing that the enhanced gradient for w_{ij} was obtained from Equation (4.7) by setting the transformations f_i and f_j to $\langle v_i \rangle_{\text{dm}}$ and $\langle h_j \rangle_{\text{dm}}$, respectively. Based on this observation, by replacing these transformations equivalently for the gradients of visible and hidden biases, the enhanced gradients for them can be derived as

$$
\begin{aligned}
b_i &\leftarrow b_i + \eta \Big[\nabla b_i - \sum_j \langle h_j \rangle_{\text{dm}} \left(\nabla w_{ij} - \langle v_i \rangle_{\text{dm}} \nabla c_j - \langle h_j \rangle_{\text{dm}} \nabla b_i \right) \Big] \\
c_j &\leftarrow c_j + \eta \Big[\nabla c_j - \sum_i \langle v_i \rangle_{\text{dm}} \left(\nabla w_{ij} - \langle v_i \rangle_{\text{dm}} \nabla c_j - \langle h_j \rangle_{\text{dm}} \nabla b_i \right) \Big]
\end{aligned}
$$

From Equations (4.2) and (4.11), it is apparent that $\nabla w_{ij} - \langle v_i \rangle_{\text{dm}} \nabla c_j - \langle h_j \rangle_{\text{dm}} \nabla b_i$ is equivalent to the enhanced gradient for the weights. Thus, replacing it with $\tilde{\nabla} w_{ij}$ yields the enhanced gradients for both visible and hidden biases (4.8) – (4.9).