2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Parallel Tempering with Equi-energy Moves for Training of Restricted Boltzmann Machines

Nannan Ji and Jiangshe Zhang

Abstract— **Training RBMs is laborious due to the difficulty of sampling from model's distribution. Although using Parallel Tempering (PT) alleviates the problem to some extent, it will result in low swap acceptance ratio when the states' energies of neighboring chains are very different. In this paper, we propose a novel PT algorithm based on the principle of swapping between chains with the same level of energy. This new algorithm partitions the state space obtained by a population of Gibbs sampling chains into several energy rings. In each ring, states have similar energies and swapping of each pair of states are conducted with a probability. Experiments on a toy dataset as well as the MNIST dataset shown that the new algorithm keeps high swap acceptance ration and results in better likelihood scores compared to several training methods.**

I. Introduction

In recent years, deep learning approaches have attracted significant interest as a way for learning layered, hierarchical representations of high-dimensional data [1]–[5]. Among the deep learning approaches, deep belief networks (DBNs) [1]–[4] are the most popular ones, and have been successfully applied to a variety of real-world applications [1]–[10]. A DBN can be viewed as a composition of several restricted Boltzmann machines (RBMs) [11]–[14] in which the hidden layer of one RBM is used as the visible layer of another RBM. By using layer-wise unsupervised pre-training of RBMs, a more accurate model for discovering the structure hidden in the data is easy to find. Therefore, it is very important to have an efficient and well-behaving learning method for RBMs.

RBM is a generative model, and its parameters can be optimized by performing stochastic gradient descent on the log-likelihood of the parameters given the training data. This gradient contains two main terms: the so-called positive phase raises the probability of training data and the so-called negative phase involves an expectation over data sampled from model's distribution. The positive phase can be obtained easily, but the negative phase is intractable since getting an unbiased sample from the model's distribution is much more difficult. Even though the Markov Chain Monte-Carlo can be used, it is time-consuming to perform alternating Gibbs sampling to converge to the equilibrium distribution of model.

Nannan Ji and Jiangshe Zhang are with School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China (email: jinannan85@gmail.com (N.N. Ji); jszhang@mail.xjtu.edu.cn (J.S. Zhang)

This work was supported by the National Basic Research Program of China (973 Program) under Grant no. 2013CB329404, the National Natural Science Foundation of China under Grant Nos. 91230101, 61075006, 11131006, 11201367, the Research Fund for the Doctoral Program of Higher Education of China under Grant No 20100201120048.

Contrastive Divergence (CD) algorithm proposed by Hinton [13] replaces the expectation with a finite set of negative samples, which are obtained by running a short Markov chain initialized at positive training data. This yields a biased, but low-variance gradient which has been shown to work well for training RBMs [1]. However, the data-centric focus of CD training can result in spurious probability modes far from the training data. To improve upon CD's limitation, the Persistent Contrastive Divergence (PCD) was proposed [15] to approximate the negative phase of gradient with samples drawn from a persistent Markov chain. This method can find a high probability region far away from the training data. However, as training progresses and the model's parameters get larger, the distribution of model becomes more rough. This will increase the chance that the chain gets stuck in local modes of the distribution. To obtain better mixing rates, Parallel Tempering (PT) was proposed [16][17], in which supplementary Gibbs chains were introduced to sample from more and more smoothed replicas of the target distribution. This algorithm is one of the most promising sampling techniques for RBM training.

Although PT achieves better mixing of the underlying chain used to generate samples from the model, great care must be taken to select the set of temperatures. Having too few or incorrectly spaced chains can result in low swap rates between neighboring chains which is disadvantageous to RBM training. In this paper, we propose an novel training algorithm for RBMs which combines PT with the principle of swapping between chains under the same level of energy. Instead of swapping states between neighboring chains, we partition the state space into several energy rings based on the energies of current states of the Gibbs sampling chains. In each energy ring, states have similar energy levels, and the swapping is conducted between chains lying in the same ring.

The rest of this paper is organized as follows: after providing a brief technical overview of RBMs and the main algorithms for training them in Section II and Section III. We describe the new algorithm in Section IV. Then, the new algorithm is compared to CD, PCD and PT on a toy dataset as well as on the MNIST dataset in Section V. In section VI, some related work is introduced. Finally, we conclude this paper with a summary in Section VII.

II. Restricted Boltzmann Machines

An RBM [11]–[14] is a two-layer, bipartite, undirected graphical model with a set of visible units $\mathbf{v}$ of dimension D representing the observable data, and a set of binary hidden

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

units $\mathbf{h}$ of dimension K learned to represent features that capture higher-order correlations in the observed data. These two layers are connected by a symmetrical weight matrix $W \in R^{D\times K}$, whereas there are no connections within the same layer. Originally, RBMs were constructed using binary visible and hidden units, but many other types of units can be used. In this paper, we focus on binary case for both layers.

Given the energy function $E(\mathbf{v},\mathbf{h})$ of the state $(\mathbf{v},\mathbf{h})$, the joint distribution over the visible and hidden units is defined by

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z} exp(-E(\mathbf{v},\mathbf{h})), \tag{1}$$

$$Z = \sum_{\mathbf{v}}\sum_{\mathbf{h}} exp(-E(\mathbf{v},\mathbf{h})). \tag{2}$$

where Z is the partition function. The energy function has the form

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{D}\sum_{j=1}^{K} v_i W_{ij} h_j - \sum_{j=1}^{K} b_j h_j - \sum_{i=1}^{D} c_i v_i, \tag{3}$$

where b_j and c_i are the hidden and visible unit biases, respectively. $\theta = \{w_{ij}, b_j, c_i\}$ includes all parameters needed to be learned.

Since RBM is a generative model, its parameters can be optimized by performing stochastic gradient descent on the log-likelihood of the parameters given the training data. The probability that the network assigns to a visible vector (training data) is given by summing over all possible hidden vectors, i.e.,

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{h} exp(-E(\mathbf{v},\mathbf{h})). \tag{4}$$

The derivative of the log probability of a training data with respect to θ is simple and can be expressed as

$$\begin{aligned}\frac{\partial \log p(\mathbf{v})}{\partial\theta} &= -\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial\theta} + \sum_{\mathbf{x}} p(\mathbf{x})\frac{\partial E(\mathbf{x})}{\partial\theta} \\ &= \langle\frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial\theta}\rangle_{p(\mathbf{h}|\mathbf{v})} - \langle\frac{\partial(-E(\mathbf{x}))}{\partial\theta}\rangle_{p(\mathbf{x})},\end{aligned} \tag{5}$$

where the angle brackets denote expectations under the distribution specified by the subscript that follows. The first term on the right side corresponds to sampling hidden configurations when the visible units are clamped to the training data; it is usually called the positive phase. The second term corresponds to obtaining joint hidden and visible samples from the current model; it is usually called the negative phase.

III. Approximating the RBM Log-likelihood Gradient

From the energy function, we can see that the hidden units h_j are independent of each other when conditioning on $\mathbf{v}$ since there are no direct connections between hidden units. Similarly, the visible units v_i are also independent of each other when conditioning on $\mathbf{h}$. In detail, the state h_j of each hidden unit j is set to be 1 with a probability

$$p(h_j = 1|\mathbf{v}) = \sigma(\sum_i W_{ij} v_i + b_j), \tag{6}$$

where $\sigma(s) = 1/(1+exp(-s))$ is the sigmoid function. The binary state v_i of each visible unit i is set to be 1 with a probability

$$p(v_i = 1|\mathbf{h}) = \sigma(\sum_j W_{ij} h_j + c_i). \tag{7}$$

Based on above analysis, it is straightforward to sample $\mathbf{h}|\mathbf{v}$ and $\mathbf{v}|\mathbf{h}$ in the RBMs, so obtaining samples for the positive phase is easy. However, obtaining samples from the model for the unclamped negative phase is not easy, as it would require running the alternating Gibbs sampling to converge to the equilibrium distribution of model. This is time-consuming.

So far, many learning algorithms have been proposed to estimate the negative phase. The most widely used methods are CD, PCD and PT.

CD is a standard way to train RBMs. Instead of approximating the negative phase in the log-likelihood gradient by a sample from the model's distribution, CD runs a Gibbs chain for only several steps which is initialized with a training data. The last sample of the chain is used to estimate the negative phase.

PCD on the other hand, approximates the gradient by drawing negative phase samples from a persistent Markov chain which are run for several Gibbs sampling steps after each parameter update. That is to say, the initial state of the current Gibbs chain is equal to the one from the previous update step, rather than the training data.

Both CD and PCD exploit a single Markov chain to estimate the negative phase. However, reliance on a single Markov chain often leads to degenerative training. When faced with the kind of multimodal target distributions, the Gibbs sampling employed in CD and PCD are subject to becoming stuck in local maxima of probability density.

PT introduces supplementary Gibbs chains that sample from more and more smoothed replicas of the original distribution. This can be formalized in the following way: given an ordered sequence of temperatures t_r from temperature $t_1 = 1$ that samples from the target distribution to a high temperature t_N, i.e., $1 = t_1 < t_2 < ... < t_N$, a set of N Markov chains was defined with stationary distributions

$$P_r(\mathbf{v},\mathbf{h}) = \frac{1}{Z_r} exp(-\frac{1}{t_r}E(\mathbf{v},\mathbf{h})), \tag{8}$$

for $r = 1, ..., N$, where $Z_r = \sum_{\mathbf{v}}\sum_{\mathbf{h}} exp(-\frac{1}{t_r}E(\mathbf{v},\mathbf{h}))$ is the corresponding partition function. Each iteration of the PT algorithm is decomposed into local and global moves. During local moves, each chain is updated independently of others. In this way, N states $(\mathbf{v}_1,\mathbf{h}_1), ..., (\mathbf{v}_N,\mathbf{h}_N)$ can be yielded. For the global move, two neighboring chains running at temperature t_r and t_{r+1} may exchange their states $(\mathbf{x}_r)$

and $(\mathbf{x}_{r+1})$ with an exchange probability given by

$$min\{1, exp((\frac{1}{t_r} - \frac{1}{t_{r+1}}) * (E(\mathbf{x}_r) - E(\mathbf{x}_{r+1})))\}, \quad (9)$$

where $\mathbf{x}_r = (\mathbf{v}_r, \mathbf{h}_r)$ and $\mathbf{x}_{r+1} = (\mathbf{v}_{r+1}, \mathbf{h}_{r+1})$. After performing these swaps between chains, sample $\mathbf{v}_1$ is taken as a sample from the model distribution.

IV. Training RBMs using Parallel Tempering with Equi-Energy Moves

PT sampling achieves good mixing rate by introducing supplementary Gibbs chains and exchanging the states of neighboring chains. The acceptance ratio for this exchange depends on the temperatures and the energies of states of neighboring chains. Based on equation (5), once the temperatures are fixed, the exchange probability only relies on the energies of two states. Specifically, the smaller the difference between two states' energies, the higher the exchange probability is. Therefore, having too few or incorrectly spaced chains in PT will result in low swap rates between neighboring chains since the states of neighboring chains may have very different energy levels. In view of this, we propose a new type of move called equi-energy move, that aims to explore the state space by moving directly between states with similar energy. Fig.1 illustrates the equi-energy move. This new algorithm is named as parallel tempering with equi-energy moves, referred to as PTEE.

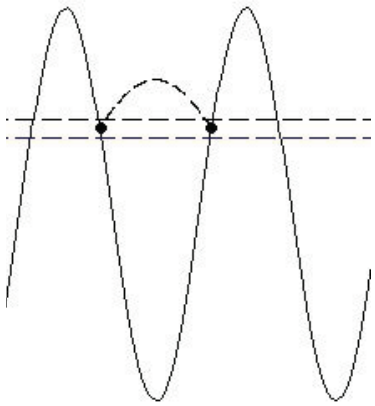

Fig. 1. Illustration of the equi-energy move, where the sampler can move freely between the states with similar energy levels.

PTEE utilizes the temperature-energy duality, and targets the energy directly. Based on a sequence of temperatures, a population of auxiliary distributions which are the smoothed versions of the target distribution is generated. In comparison with the target distribution, these versions are easier to sample from as they exhibit greater ergodicity. This kind of technique by using multiple auxiliary distributions mitigates the shortcoming of the single Markov chain of easily being stuck in local maxima of probability density. By updating each chain via classical MCMC algorithms, the current states of each chain can be obtained. PTEE then partitions the state space into several equi-energy set in terms of the states' energies of all chains. The new type of move used in PTEE, i.e. equi-energy move, encourages the current state to move to another state draw from the already constructed equi-energy set that has energy level close to the current state. This equi-energy move guarantees high swap acceptance ratio since the energy levels of states used to swap are similar, making it easy to render the diversity of states captured by the target distribution. In a ward, PTEE facilitates global moves between different chains, resulting in a good exploration of the state space by the target chain. Details of this algorithm will be described as follows.

First, a sequence of $d+1$ energy levels is introduced:

$$H_1 < H_2 < ... < H_{d+1} = \infty, \quad (10)$$

such that H_1 is below the minimum energy, i.e., $H_1 \leq min(E(\mathbf{x}))$. Associated with the energy levels is a sequence of temperatures

$$1 = t_1 < ... < t_N. \quad (11)$$

PTEE also considers a population of N Markov chains associated with probability $p_r(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_r} exp(-\frac{1}{t_r} E(\mathbf{v}, \mathbf{h}))$, where $Z_r = \sum_{\mathbf{v}} \sum_{\mathbf{h}} exp(-\frac{1}{t_r} E(\mathbf{v}, \mathbf{h}))$ is the corresponding partition function.

In each step of PTEE, both local moves and global moves involve obtaining the samples from RBM model. During the local moves, we run N Markov chains based on the Gibbs sampling from N distribution, $p_r(r = 1, ..., N)$, in which each chain is locally updated independently of others. The obtained states of these N chains are denoted as $(\mathbf{v}_1, \mathbf{h}_1), (\mathbf{v}_2, \mathbf{h}_2), ..., (\mathbf{v}_N, \mathbf{h}_N)$. When we get the states of N chains, the exchanges of these states will be conducted to improve mixing rate. These exchanges are called global moves. In the process of global moves, the states of N Markov chains are firstly partitioned into several energy rings according to the states' energies and $d+1$ energy levels. Each energy ring $D_j (j = 1, ..., d)$ is constructed as follows:

$$D_j = \{(\mathbf{v}, \mathbf{h}) : E(\mathbf{v}, \mathbf{h}) \in [H_j, H_{j+1})\}, j = 1, ..., d. \quad (12)$$

Then, the swapping is carried out in the interior of energy rings which contain at least two chains. In detail, in the interior of each energy ring containing at least two chains, each pair of chains exchange their current states with a probability given by

$$min\{1, exp((\frac{1}{t_r} - \frac{1}{t_s}) * (E(\mathbf{v}_r, \mathbf{h}_r) - E(\mathbf{v}_s, \mathbf{h}_s)))\}, \quad (13)$$

where r and s respectively index two chains lying in the same ring. In practice, the swapping mentioned above can be executed from the chain with the highest temperature to the one with the lowest temperature.

After performing the swaps, we take the (eventually exchanged) samples $\mathbf{v}_1$ of the original chain (with temperature $t_1 = 1$) as a sample from the RBM distribution. This procedure is repeated L times yielding samples $\mathbf{v}_{1,1}, \mathbf{v}_{1,2}, ..., \mathbf{v}_{1,L}$ used for the approximation of the expectation under the RBM distribution in the gradient of log-likelihood.

The levels $H_1, H_2, ..., H_d$ are critical since these levels determine the energy rings. Having proper energy levels can achieve high swap probability. In order to cover the energies of states which need to be partitioned, we set H_1 as the minimum energy of these states, and set H_d as the maximum energy. Once the values of H_1 and H_d are chosen, many approaches can be used to yield $H_2, ..., H_{d-1}$. For example, $H_2, ..., H_{d-1}$ can be obtained so that $ln(H_{r+1} - H_r)$ or $(H_{r+1} - H_r)$ are evenly spaced. In this paper, we set the $(H_{r+1} - H_r)$ to be evenly spaced. Here, it should be noticed that the values of the RBM's parameters are dynamic during the training procedure. This leads to a dynamic energy range of the states that need to be partitioned. Therefore, in each step of the parameters' update, we set the corresponding energy levels based on the current states of N chains: the minimum energy of these states is set to be the value of H_1; the maximum energy is set to be the value of H_d. That is to say, the energy levels are dynamic. In practical implement, we only need to set the number of energy rings (i.e., d) before training RBM. The corresponding energy levels can be obtained by the method mentioned above.

When the number of energy ring is 1 and the energy levels of all steps of updates are set to be $-\infty$ and ∞, PTEE degenerates to PT.

The main idea of PTEE seems to be close to another algorithm proposed by Baragatti et al. which also uses equi-energy move in parallel tempering, but these two algorithms have many differences. First of all, PTEE applies the equi-energy move to a more specifical and more important problem, the training of RBMs. Secondly, according to the training process of RBMs, PTEE utilizes dynamic energy levels rather than static energy levels to yield energy rings. In the training process of RBMs, the values of model parameters are constantly changing, which leads to dynamic energy range of states. Under this circumstance, if we use the static energy levels, it will happen that lots of states are attributed to the same ring in many steps of updates, reducing the advantages of equi-energy move. Thirdly, PTEE makes equi-energy moves in all energy rings that contain at least two chains, rather than one ring that is chosen randomly. Last, in each energy ring that has been chosen, PTEE takes into account the swaps of each pair of states from the chain with the highest temperature to the one with the lowest temperature, rather than two chains that are chosen uniformly in the energy ring.

V. Related Work

The equi-energy sampler has been proposed by Kou et al. [18] based on a population of chains. In this approach, the distribution of Markov chain is obtained by the energy truncation which increases the difficulty of using Gibbs sampling. This difficulty comes from the piecewise of conditional distribution $p(v_i = 1|\mathbf{h})$ or $p(h_j = 1|\mathbf{v})$, in which the energy of state $(\mathbf{v}, \mathbf{h})$ needs to be calculated in the case of incomplete information about the state $(\mathbf{v}, \mathbf{h})$. Baragatti et al. [19] proposed another equi-energy sample algorithm which combines PT with the equi-energy moves. The differences between Baragatti's method and our approach have been mentioned in above section.

VI. Experiments

In this section, we evaluated the performance of PTEE algorithm and compare it with CD, PCD and PT. In order to compute the exact log-likelihood of our model, we firstly carried out experiments on a toy dataset. Then experiments on the MNIST dataset were also executed to validate the efficiency of the proposed algorithm.

A. Toy dataset

The toy dataset is composed of 10,000 images, each image with 4×4 binary pixels. The way for generating this dataset is same as the one mentioned in [17]. Specifically, four basic modes chosen to be maximally distant from one another are used to construct this dataset. For each basic mode, we generated 2,500 near replicas by considering a probability 0.001 of permuting each pixel independently. Some randomly chosen samples are shown in Fig.2.

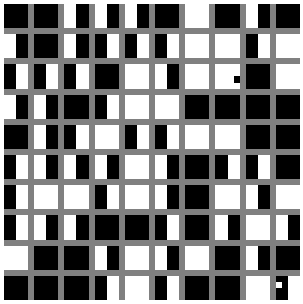

Fig. 2. Some random samples from our toy dataset.

For all comparative algorithms, we exploited online learning and performed 300,000 weight updates for both positive and negative phases. This means that each training sample is presented 30 times to each model. The number of hidden unit is set to be 10. The learning rate is decreased linearly towards zero since we found by some preliminary experiments that a decreasing learning rate schedule was necessary for the model to learn a good generative model. No weight decay was used in any of the models. For the other hyperparameters, we tested the initial learning rates in $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. The tempered models were trained with $\{5, 7, 10\}$ parallel Markov chains with minimal inverse temperature of 0. For CD and PCD, the number of Gibbs sampling steps between consecutive parameter updates was set to be 5, 7 and 10 to offset the computational cost of having parallel chains. Tempered models were limited to a single Gibbs step between consecutive parameter updates. In addition, for PTEE, the number of energy rings was tested in $\{3, 5, 7, 10\}$.

Based on some preliminary experiments, we find that using 10 chains achieves better likelihood scores compared to 5 and 7 chains. So, we show the results obtained by using PTEE and PT with 10 chains, as well as CD and PCD with 10

Gibbs sampling steps between consecutive parameter updates in Fig.3. These results are the averaged log-likelihood by training each model 5 times. It can be seen that PTEE is able to yield good result more quickly than PT, and ultimately obtain better likelihood score. This demonstrates that PTEE improves the mixing rate. In the mean time, we also found that using single Gibbs chain is unable to cope with the model diverges. With the increase of the update steps, the mixing of CD and PCD degrades, resulting in a sudden drop in likelihood.

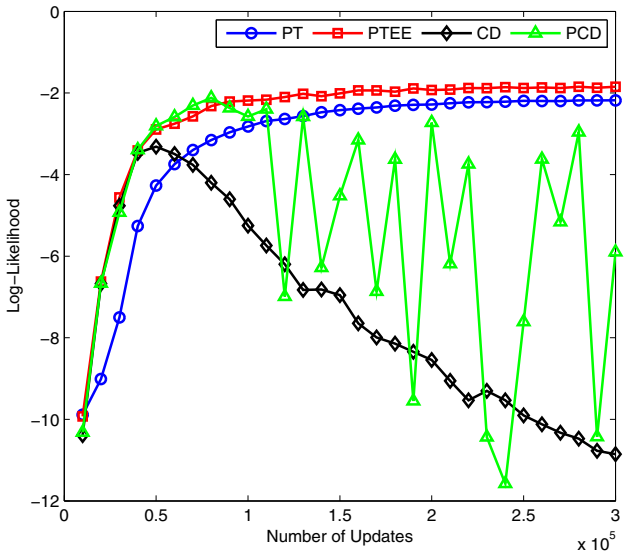

Fig. 3. Log-likelihood of the training data against the number of updates over the toy dataset. The number of chains is 10.

In Fig.4, we also show the result obtained using PTEE with 5 chains. PT10 represents the PT algorithm with 10 chains. PTEE10 and PTEE5 respectively denote the PTEE algorithm with 10 chains and 5 chains. From this figure, it can be seen that PTEE still yield a good result similar to one using 10 chains when we use 5 chains, which means that PTEE is immune to the number of Markov chains to some extent.

Next, we investigated the difference of global moves between PT and PTEE. We first show the energies of states, among which each pair of states are considered to be swapped with a probability. For PT, these states are all chains' states under one update step, as shown in the top figure of Fig.5. For PTEE, these states belong to the same energy ring under one update step, which is shown in the same color in the bottom figure of Fig.5. From Fig. 5, we can observe that the states of neighboring chains obtained by PT algorithm may have very different energies, which leads to low swap rate. While the states belonging to the same energy ring obtained by PTEE have similar energies, which ensures high swap rates.

Fig. 6 shows the average probability of swaps of each chain, which can be calculated by equations (9) and (13).

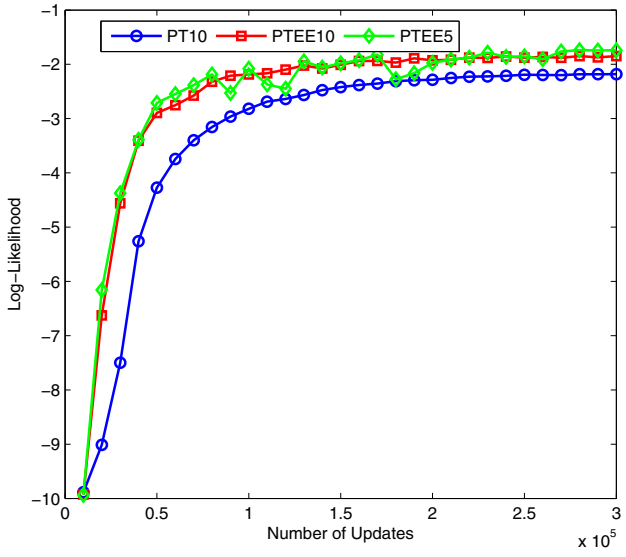

Fig. 4. Log-likelihood of the training data against the number of updates over the toy dataset. The number of chains is 5 or 10.

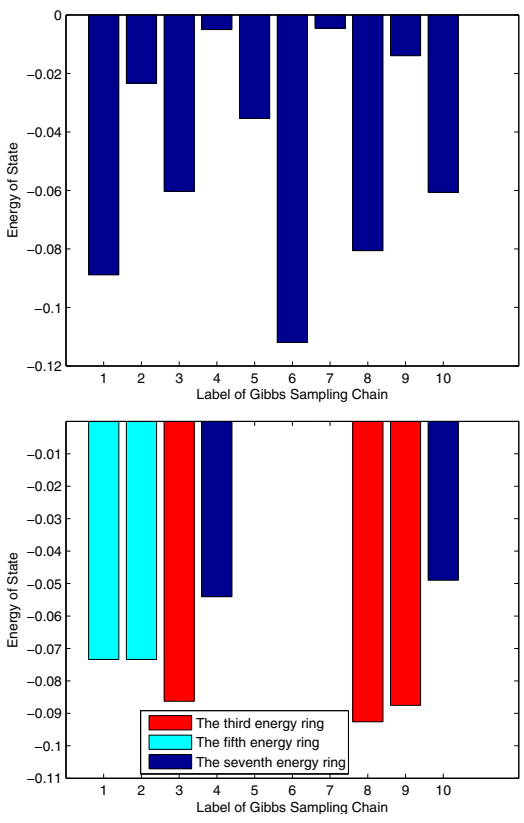

Fig. 5. The energies of states obtained by experimenting on the toy dataset: the top figure shows the energies of states obtained by PT under one update step; the bottom figure shows the energies of states belonging to the same energy ring containing at least two chains obtained by PTEE under one update step.

Table 1: The number of swaps between any two chains obtained by PT10 over the toy dataset. PT10 represents PT algorithm with 10 chains.

Gibbs sampling chain	chain 1	chain 2	chain 3	chain 4	chain 5	chain 6	chain 7	chain 8	chain 9	chain 10
chain 1	0	113909	0	0	0	0	0	0	0	0
chain 2	113909	0	138476	0	0	0	0	0	0	0
chain 3	0	138476	0	148781	0	0	0	0	0	0
chain 4	0	0	148781	0	164389	0	0	0	0	0
chain 5	0	0	0	164389	0	186086	0	0	0	0
chain 6	0	0	0	0	186086	0	206910	0	0	0
chain 7	0	0	0	0	0	206910	0	222331	0	0
chain 8	0	0	0	0	0	0	222331	0	234624	0
chain 9	0	0	0	0	0	0	0	234624	0	243791
chain 10	0	0	0	0	0	0	0	0	243791	0

Table 2: The number of swaps between any two chains obtained by PTEE10 over the toy dataset. PTEE10 represents PTEE algorithm with 10 chains.

Gibbs sampling chain	chain 1	chain 2	chain 3	chain 4	chain 5	chain 6	chain 7	chain 8	chain 9	chain 10
chain 1	0	113724	5336	1609	681	336	165	128	96	62
chain 2	113724	0	91362	12515	2387	645	284	166	103	89
chain 3	5336	91362	0	71759	15245	3208	872	339	201	112
chain 4	1069	12515	71759	0	62697	19362	5614	1690	567	261
chain 5	681	2387	15245	62697	0	68289	26225	8896	2934	986
chain 6	336	645	3208	19362	68289	0	86555	31348	10040	3327
chain 7	165	284	872	5614	26225	86555	0	107443	32725	10529
chain 8	128	166	339	1690	8896	31348	107443	0	101204	44230
chain 9	96	103	201	567	2934	10040	32725	101204	0	36794
chain 10	62	89	112	261	986	3327	10529	44230	36794	0

We can see that almost all chains of PTEE have higher swap probabilities compared with PT. In addition, we also tested the number of swaps between any two chains. The corresponding results are displayed in Table 1 and Table 2. For PT, only neighboring chains swap their states. While for PTEE, the states of any two chains are swapped.

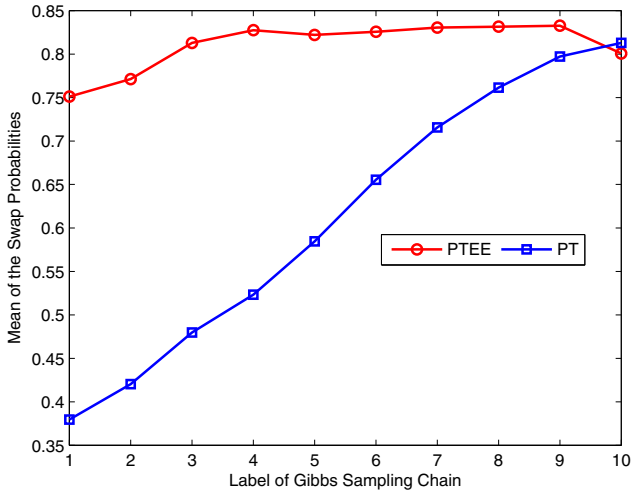

Fig. 6. Average values of swap probabilities corresponding to each chain over the toy dataset.

Moreover, the dataset generated by setting p to be 0.1 was also used to test the efficiency of PTEE. The experimental conditions are same as ones used in the previous experiments. Fig. 7 displays the average likelihood scores obtained by training each model 5 times. From this figure, we can see that PTEE still outperforms PT and other comparative algorithms when p is set to be a bigger value.

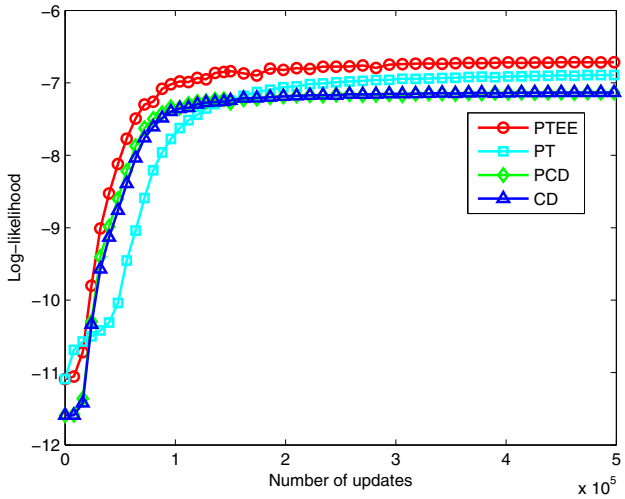

Fig. 7. Log-likelihood of the training data against the number of updates over the toy dataset generated by setting p to be 0.1. The number of chains is 10.

B. MNIST dataset

The MNIST dataset is more complicated and higher dimensional than the above toy dataset, which contains 60,000

training and 10,000 test images of ten handwritten digits (0 to 9), each image with 28×28 pixels [20]. In our experiments, considering the computational cost, we randomly sampled 1,000 images per class from the 60,000 training images to compare the efficiency among PTEE and the comparative algorithms. The pixel intensities were scaled between 0 and 1 and interpreted as probabilities from which binary values were sampled.

The size of mini-batches and the number of hidden units are respectively set to be 100 and 500. Here, the number of data's dimension and model's hidden units are very big. So, we cannot compute the exact log-likelihood of RBM models since the normalizing constant cannot be exactly calculated. In view of this, we adapts the method utilizing annealed importance sampling to estimate the normalizing constant, which has been successfully adapted for computing the normalizing constant of RBMs [21].

The setups of learning rate and weight initialization as well as sampling or not sampling in the negative phase are same as ones used in the experiments of toy dataset, except the number of parameter updates. Here, we performed 100,000 weight updates for both positive and negative phases. Considering that PT is the most competitive model with PTEE, we only compare these two models on the MNIST dataset. Fig. 8 shows the average likelihood scores by training RBMs via PTEE and PT 5 times. It can be observed that PTEE works better than PT.

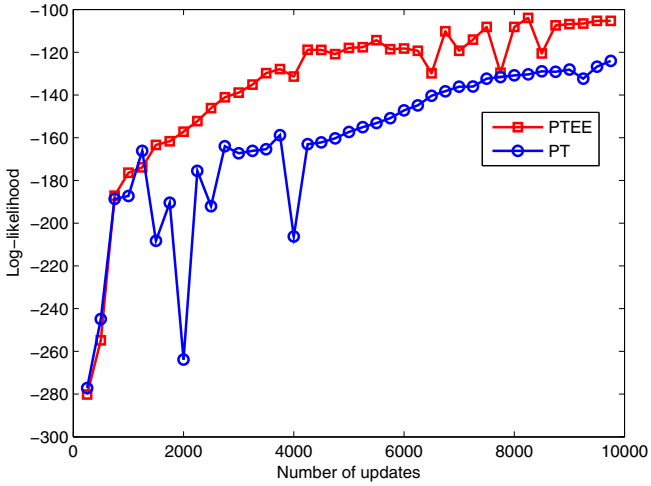

Fig. 8. Log-likelihood of the training data against the number of updates over the MNIST dataset. The number of chains is 10.

C. Analysis of the time complexity

Here, we want to briefly discuss the time complexity of the considered several algorithms. CD and PCD learning algorithms are divided into three phases: positive phase which samples the hidden units when the visible units are clamped on training data; negative phase which computes the reconstruction and samples hidden units when the visible units are clamped on the reconstruction; and update phase. Suppose that the number of training samples is m, D and K respectively denote the number of visible and hidden units. The time complexities of positive and negative phases are $O(m \times D \times K)$. And the time complexity of update phase is $O(D \times K)$. Therefore, the overall time complexity of CD and PCD training with G steps Gibbs sampling are $O(T \times m \times D \times K \times G)$, where T is the number of epochs. For PTEE and PT, except for the above three phases, several auxiliary Markov chains are exploited and a swapping phase was also used. Suppose the number of Markov chains is C. The time complexities of PTEE and PT are respectively $O(T \times m \times D \times K \times C)$. Here, PTEE and PT are limited to a single Gibbs step between consecutive parameter updates.

Moreover, we also recorded the running times of different algorithms on one epoch and on one training sample when they are applied to the toy dataset and the MNIST dataset. Here, the number of Gibbs sampling steps between consecutive parameter updates in CD and PCD is 10. The number of parallel Markov chains in PTEE and PT is 10. And a single Gibbs step was used in PTEE and PT between consecutive parameter updates. All experiments were conducted on a Windows machine with Inter(R) Core(TM) i7-2600 3.40GHz CPU and 8GB RAM. Table 3 displays the results. Based on the results, we can see that the running times of PTEE is bit less than PT since not all of the adjacent chains need to exchange their states in PTEE.

Table 3: The running times (in seconds) of different algorithms when they are applied to toy dataset and MNIST dataset.

Dataset	CD	PCD	PT	PTEE
chain 1	0.00912	0.00937	0.01161	0.01027
chain 2	31.03092	31.17306	31.33132	31.28487

VII. Conclusions

In this paper, a new algorithm for training RBMs was proposed, which combines parallel tempering and equi-energy sampler. Given the number of energy levels, the new algorithm yields dynamic energy rings. Thanks to relevant equi-energy moves in the same ring, the proposed algorithm achieves good mixing of the generated Markov chains. Experiments on a toy dataset as well as the MNIST dataset demonstrate that this new algorithm achieves better likelihood scores more quickly than CD, PCD and PT. Meanwhile, the performance is immune to the number of chains to some extent since equi-energy moves ensure high swap probability.

References

[1] G.E. Hinton, S. Osindero, Y. Teh, "A fast learning algorithm for deep belief nets", *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

[2] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, "Greedy layer-wise training deep networks", *Advances in Neural Information Processing Systems*, 2007, pp. 153-160.

[3] M. Ranzato, C. Poultney, S. Chopra, Y. LeCun, "Efficient learning of sparse representations with an energy-based model", *Advances in Neural Information Processing Systems*, 2006, pp. 1137-1144.

[4] G.E. Hinton, R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", *Science*, vol. 313, no. 5786, pp. 504-507, 2006.

[5] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation", *the 24th International Conference on Machine Learning*, 2007, pp. 473-480..
[6] R. Salakhutdinov, G. Hinton, "Semantic hashing", *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969-978, 2009.
[7] G. Hinton, "To recognize shapes, first learn to generate images", *Progress in Brain Research*, vol. 165, pp. 535-547, 2007.
[8] T. Deselaers, S. Hasan, O. Bender, H. Ney, "A deep learning approach to machine transliteration", *the 4th Workshop on Statistical Machine Translation*, 2009, pp. 233-241.
[9] E. Horster, R. Lienhart, "Deep networks for image retrieval on large-scale databases", *the 16th ACM International Conference on Multimedia*, 2008, pp. 643-646.
[10] E. Chen, X. Yang, H. Zha, R. Zhang, W. Zhang, "Learning object classes from image thumbnails through deep neural networks", *the International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 829-832.
[11] A. Fischer, C. Igel, "An introduction to restricted Boltzmann machines", *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 2012, pp. 14-36.
[12] A. Fischer, C. Igel, "Training restricted Boltzmann machines: an introduction", *Pattern Recognition*, vol. 47, no. 1, pp. 25-39, 2014.
[13] G.E. Hinton, "Training products of experts by minimizing contrastive divergence", *Neural Computation*, vol. 14, no. 8, pp. 1711-1800, 2002.
[14] Y. Freund, D. Haussler, "Unsupervised learning of distributions on binary vectors using two layer networks", *Advances in Neural Information Processing Systems*, 1992, pp. 912-919.
[15] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient", *the 25th International Conference on Machine Learning*, 2008, pp. 1064-1071.
[16] K.H. Cho, T. Raiko, A. Ilin, "Parallel tempering is efficient for learning restricted boltzmann machines", *the 2010 International Joint Conference on Neural Networks*, 2010, pp. 1-8.
[17] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, O. Dellaleau, "Parallel tempering for training of restricted boltzmann machines", *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 145-152.
[18] S.C. Kou, Q. Zhou, W.H. Wong, "Discussion paper equi-energy sampler with applications in statistical inference and statistical mechanics", *The annals of Statistics*, vol. 34n no. 5, pp. 1581-1619, 2006.
[19] M. Baragatti, A. Grimaud, D. Pommeret, "Parallel tempering with equi-energy moves", *Statistics and Computing*, vol. 23, pp. 323-339, 2013.
[20] Y. LeCun, The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist.
[21] R. Salakhutdinov, "Learning deep generative models", Ph.D. dissertation, Massachusetts Institute of Technology, 2009.