Parallel Tempering is Efficient for Learning Restricted Boltzmann Machines

KyungHyun Cho, Tapani Raiko, Alexander Ilin

***Abstract*—A new interest towards restricted Boltzmann machines (RBMs) has risen due to their usefulness in greedy learning of deep neural networks. While contrastive divergence learning has been considered an efficient way to learn an RBM, it has a drawback due to a biased approximation in the learning gradient. We propose to use an advanced Monte Carlo method called parallel tempering instead, and show experimentally that it works efficiently.**

I. Introduction

Recently, deep neural networks such as a deep belief network [1] and a deep Boltzmann machine [2] have become widely applied to various machine learning tasks. These deep neural networks are characterized and distinguished from the conventional multi-layer perceptron and other shallow neural networks by their large number of layers of neurons and its adaptation by a layer-wise unsupervised greedy learning method.

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 which is situated one level up. This way of constructing deep neural networks allows for using layer-wise training of RBMs, which facilitates finding a more accurate model for the data. Such multi-stage learning has been empirically shown to work better than conventional learning methods such as the widely used back-propagation [2]. It is thus important to have an efficient and well-behaving learning method for RBM. In this paper, we explore some advanced sampling procedures for that.

The paper starts by briefly discussing theoretical background behind a general Boltzmann machine (BM). Then, we state the difference between the general BM and RBM, and introduce a learning algorithm specific to RBM. Contrastive divergence learning which is a successful learning algorithm for RBM is explained, and later a learning algorithm utilizing parallel tempering is introduced. In the experimental part, we demonstrate the capability of RBM to learn the data distribution by drawing samples from a trained RBM. We also compare the newly proposed learning algorithm with contrastive divergence learning.

The authors are with the Department of Information and Computer Science, School of Science and Technology, Aalto University, Finland (email: firstname.lastname@tkk.fi)

II. Boltzmann Machine

Boltzmann machine (BM) is a stochastic recurrent neural network consisting of binary neurons [3]. The network is fully connected, and we use 0 or 1 as the state of each neuron x_i. The links between each pair of neurons are symmetric (meaning that the effect of one neuron on the state of the other one is symmetric for each pair) and it is assumed that there are no edges going from the neurons to themselves.

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

$$E(\mathbf{x} \mid \mathbf{W}) = -\frac{1}{2}\sum_i \sum_{j>i} w_{ij} x_i x_j,$$

where $\mathbf{W}$ is a weight matrix consisting of weights w_{ij} of the synaptic connections between neurons i and j. We assume that $w_{ii} = 0$ and that $w_{ij} = w_{ji}$. The bias terms can be omitted by using an auxiliary component in the state vector that always has the value 1. The probability of a state $\mathbf{x}$ is

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

where

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

is the normalizing constant.

It follows from (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:

$$P(x_i = 1 \mid \mathbf{x}_{\setminus i}, \mathbf{W}) = \frac{1}{1 + \exp\left(-\sum_{j \neq i} w_{ij} x_j\right)}, \tag{2}$$

where $\mathbf{x}_{\setminus i}$ denotes a vector $[x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_d]^T$. It is obvious that this probability is expressed using the standard nonlinear sigmoid function used in multi-layer perceptron networks.

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.

A. Learning Boltzmann machine

The parameters of BM can be learnt 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

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

where the states $\mathbf{h}$ of the hidden neurons have to be marginalized out. This yields

$$\begin{aligned}\mathcal{L}(\mathbf{W}) &= \sum_{t=1}^{N} \log \frac{\sum_{\mathbf{h}} \exp(\frac{1}{2}\mathbf{x}^{(t)^T}\mathbf{W}\mathbf{x}^{(t)})}{\sum_{\mathbf{x}} \exp(\frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x})} \\ &= \sum_{t=1}^{N} \left[\log \sum_{\mathbf{h}} \exp(\frac{1}{2}\mathbf{x}^{(t)^T}\mathbf{W}\mathbf{x}^{(t)}) \right. \\ &\qquad \left. - \log \sum_{\mathbf{x}} \exp(\frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x}) \right].\end{aligned}$$

The gradient of the log-likelihood is obtained by taking partial derivative with respect to parameters w_{ij}

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_{ij}} &= \frac{1}{2} \sum_{t=1}^{N} \left[\frac{\sum_{\mathbf{h}} x_i^{(t)} x_j^{(t)} \exp(\frac{1}{2}\mathbf{x}^{(t)^T}\mathbf{W}\mathbf{x}^{(t)})}{\sum_{\mathbf{h}} \exp(\frac{1}{2}\mathbf{x}^{(t)^T}\mathbf{W}\mathbf{x}^{(t)})} \right. \\ &\qquad \left. - \frac{\sum_{\mathbf{x}} x_i x_j \exp(\frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x})}{\sum_{\mathbf{x}} \exp(\frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x})} \right] \\ &= \frac{N}{2} \left[\frac{1}{N} \sum_{t=1}^{N} \sum_{\mathbf{h}} x_i^{(t)} x_j^{(t)} P(\mathbf{h} \mid \mathbf{v}^{(t)}, \mathbf{W}) \right. \\ &\qquad \left. - \sum_{\mathbf{x}} x_i x_j P(\mathbf{x} \mid \mathbf{W}) \right] \\ &= \frac{N}{2} \left[\langle x_i x_j \rangle_{P(\mathbf{h}|\{\mathbf{v}^{(t)}\},\mathbf{W})} - \langle x_i x_j \rangle_{P(\mathbf{x}|\mathbf{W})} \right],\end{aligned}$$

where we used a shorthand notation $\langle \cdot \rangle_{P(\cdot)}$ which can be understood as the expectation computed over the probability distribution $P(\cdot)$.

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

$$w_{ij} \leftarrow w_{ij} + \eta \left[\langle x_i x_j \rangle_{P(\mathbf{h}|\{\mathbf{v}^{(t)}\},\mathbf{W})} - \langle x_i x_j \rangle_{P(\mathbf{x}|\mathbf{W})} \right], \quad (3)$$

where η denotes the learning rate, and it has absorbed the factor $\frac{N}{2}$. Thus the direction of the gradient is the difference between the correlations under two distinct probability distributions. The first one $\langle x_i x_j \rangle_{P(\mathbf{h}|\{\mathbf{v}^{(t)}\},\mathbf{W})}$ is the correlation obtained while the visible nodes are clamped to the training data, and it represents the target probability distribution to which the trained BM is intended towards. The second term $\langle x_i x_j \rangle_{P(\mathbf{x}|\mathbf{W})}$ is the correlation obtained from the current probability distribution represented by the BM. According to the sign of each term, the computations of the two terms can be called the positive phase and the negative phase, respectively.

The learning algorithm can be seen as driving BM so that the correlations between each pair of neurons in the two phases coincide with each other. In other words, we want the probability distribution represented by BM to be exactly identical to the probability distribution defined by the training data set. Although the analytical formulation of the exact probability distribution of the training data set is unknown, the positive phase mimics it by clamping the visible neurons to the training data, and lets the hidden neurons freely have their own states. It can then be compared to the probability distribution represented by BM where both the visible and hidden neurons can freely choose their states according to the distribution determined by the weights. When the difference between the distributions in the two phases becomes zero or small enough, then learning effectively stops.

B. Practical limitation and approximate approach

Although the activation and learning rules of BM are both clearly formulated, there are practical limitations in using BM. Especially, the gradient-based update formula is computationally unfeasible, as the distributions required in both phases can only be obtained by evaluating all possible combinations of the states of the neurons in the machine.

There exist exponentially many possible combinations of the states. For example, BM which was designed to handle 28×28 black-and-white image with 500 hidden nodes has $2^{28 \cdot 28 + 500}$ possible combinations, and the number is unimaginably huge. Evaluating all those states at every gradient update step is simply unfeasible. In fact, even the evaluation of a probability of a single combination is almost impossible, as all combinations of the states must be evaluated regardlessly to compute the normalizing constant $Z(\mathbf{W})$.

The obvious approach to overcome this difficulty is to use Gibbs sampling (see e.g. [4]). 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).

This approach can greatly reduce the computational burden of the gradient update rule. If we assume 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 number of all possible combinations of the states of the neurons, the learning of BM is not anymore as computational unfeasible as the exact computation of the probability masses.

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 [5]: Due to the nature of component-wise sampling, the samples might miss some modes of the distribution.

III. Restricted Boltzmann Machine

To overcome these practical limitations imposed on the general Boltzmann machine, a structurally restricted version of Boltzmann machine called Restricted Boltzmann Machine (RBM) has been proposed [6]. 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, and it resembles the structure of the bi-partite graph.

The most important advantage over the general BM is on the improved effectiveness in doing sampling. It follows from the fact that all neurons in one layer are independent of each other given the states of the neurons in the other layer. Now Gibbs sampling can be done layer-wise rather than component-wise. It can then greatly reduce the number of sampling runs required to get enough samples to represent the probability distribution.

Furthermore, in the view of computational efficiency, the layer-wise sampling can fully utilize the modern parallelized computing environment, as sampling of each neuron (or component) in the same layer can be done independently of each other and simultaneously, whereas Gibbs sampling on the general BM requires that sampling of each neuron must be done sequentially.

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

$$E(\mathbf{v}, \mathbf{h} \mid \mathbf{\Theta}) = -\frac{1}{2}\left(\mathbf{v}^T\mathbf{W}\mathbf{h} + \mathbf{b}^T\mathbf{v} + \mathbf{c}^T\mathbf{h}\right)$$
$$P(\mathbf{v}, \mathbf{h} \mid \mathbf{\Theta}) = \frac{1}{Z(\mathbf{\Theta})}\exp\left\{-E(\mathbf{v}, \mathbf{h} \mid \mathbf{\Theta})\right\},$$

where now parameters $\mathbf{\Theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$ include biases $\mathbf{b}$ and $\mathbf{c}$. The learning rules then become

$$w_{ij} \leftarrow w_{ij} + \eta_w\left[\langle v_i h_j\rangle_{P(\mathbf{h}|\{\mathbf{v}^{(t)}\},\mathbf{\Theta})} - \langle v_i h_j\rangle_{P(\mathbf{v},\mathbf{h}|\mathbf{\Theta})}\right]$$
$$b_i \leftarrow b_i + \eta_b\left[\langle v_i\rangle_{P(\mathbf{h}|\{\mathbf{v}^{(t)}\},\mathbf{\Theta})} - \langle v_i\rangle_{P(\mathbf{v},\mathbf{h}|\mathbf{\Theta})}\right]$$
$$c_j \leftarrow c_j + \eta_c\left[\langle h_j\rangle_{P(\mathbf{h}|\{\mathbf{v}^{(t)}\},\mathbf{\Theta})} - \langle h_j\rangle_{P(\mathbf{v},\mathbf{h}|\mathbf{\Theta})}\right],$$

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

For computing the correlations during the positive phase, the exact computation of the correlation between the clamped neurons $\mathbf{v}$ and the (free) hidden neurons $\mathbf{h}$ is possible, since all the hidden neurons are independent of each other conditioned on the training data. For instance, the expectation for updating the weights is

$$\langle v_i h_j\rangle_{P(\mathbf{h}|\{\mathbf{v}^{(t)}\},\mathbf{\Theta})} = \frac{1}{N}\sum_{t=1}^{N} v_i^{(t)} P(h_j^{(t)} \mid \mathbf{v}^{(t)}, \mathbf{\Theta}), \quad (4)$$

where $P(h_j^{(t)} \mid \{\mathbf{v}^{(t)}\}, \mathbf{\Theta})$ is computed using (2) with the bias term included.

Although the difficulties in learning have partly been solved, the practical limitations of the general Boltzmann machine still remain. 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. The computational load has been greatly reduced but it still remains large. Moreover, the problem of the multi-modal probability distribution has not at all been addressed.

A. Contrastive divergence learning

Contrastive divergence (CD) learning [7] does not follow the gradient obtained by the maximum likelihood criterion. Rather, CD learning approximates the true gradient by replacing the expectation over $P(\mathbf{v}, \mathbf{h} \mid \mathbf{\Theta})$ with an expectation over a distribution P_n that is obtained by running n steps of Gibbs sampling from the empirical distribution. In practice, parallel chains of Gibbs sampling are run starting separately from each observation in the data set. The samples at step n are used to compute the expectation.

The learning formula, then, becomes

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

It should be noted that the case $n = 0$ produces the empirical distribution $P(\mathbf{h} \mid \{\mathbf{v}^{(t)}\}, \mathbf{\Theta})$ used in the positive phase, whereas the case $n = \infty$ produces the true distribution of the negative phase $P(\mathbf{x} \mid \mathbf{\Theta})$.

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 [8]. Nevertheless, CD learning has been proven to work well in practice. A good property of CD is that in case the data distribution is multimodal, running the chains starting from each data sample guarantees, that the samples approximating the negative phase have representatives from different modes.

IV. Restricted Boltzmann Machine and Parallel Tempering

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

1) Create a sequence of RBMs $(R_0, R_1, \cdots, R_K)$ such that parameters of R_k are $\mathbf{\Theta}_k = (T_k\mathbf{W}, \mathbf{b}, \mathbf{c})$, where $0 \leq T_0 < T_1 < \cdots < T_K = 1$.
2) Create an empty set of samples $\boldsymbol{X} = \{\}$.
3) 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).
4) For m = 1 to M, repeat
   a) 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 $\mathbf{x}_{k,m-1}$.
   b) For j = K to 1, repeat
      • 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 (5).
   c) Add $\mathbf{x}_{K,m}$ to $\boldsymbol{X}$.
5) $\boldsymbol{X}$ is the set of samples collected by parallel tempering sampling.

TABLE I: Sequence of steps for sampling from RBM using parallel tempering.

state space. This paper, therefore, proposes to use yet another, improved variant of Markov-Chain Monte Carlo sampling method called *parallel tempering* (PT) [9]. PT sampling used in this paper utilizes multiple Gibbs sampling chains with varying levels of temperatures, where a term *temperature* denotes the level of the energy of the overall system, in this case, RBM. The higher the temperature of the chain, the more likely the samples collected by Gibbs sampling to move freely.

The use of PT sampling in training RBM is simply to use it instead of Gibbs sampling in the negative phase. 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 done well even with a smaller number of samples than those required if Gibbs sampling is used.

The basic idea of PT sampling is that samples are collected from multiple chains of Gibbs sampling with different temperatures from the highest temperature $T = 0$ to the current temperature $T = 1$[1]. 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. [4]) as

$$P_{\text{swap}}(\mathbf{x}_{T_1}, \mathbf{x}_{T_2}) = \min\left(1, \frac{P_{T_1}(\mathbf{x}_{T_2})P_{T_2}(\mathbf{x}_{T_1})}{P_{T_1}(\mathbf{x}_{T_1})P_{T_2}(\mathbf{x}_{T_2})}\right), \quad (5)$$

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. $P_T(\cdot)$ is the probability function of the RBM with parameters $\mathbf{\Theta} = (T\mathbf{W}, \mathbf{b}, \mathbf{c})$, where the effect of different temperatures is emulated by multiplying the current weights $\mathbf{W}$ by the corresponding temperature T.

After each round of sampling and the swaps, the sample at the true temperature $T = 1$ is gathered as the sample for the iteration. The samples come from the true distribution $P(\mathbf{v}, \mathbf{h} \mid \mathbf{\Theta})$ 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 multimodal. So, the samples from the chain are less prone to missing some modes that exist in RBM. 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.

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. A brief description of how PT sampling can be done for training RBM is shown in Table I. This is the procedure that is run between each parameter update during learning.

V. Estimating Log-Likelihood of Restricted Boltzmann Machines

For estimating the normalizing constant, this paper adapts the method utilizing *annealed importance sampling* (AIS) [10] which has been successfully adapted for computing the normalizing constant of RBM [11].

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 $P_A(\cdot)$:

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

[1]Since the lower value denotes the higher temperature, a term *inverse temperatures* is frequently used, but in this paper, *temperature* will be used.

1) Create a sequence of intermediate temperatures T_k such that $0 \leq T_0 < T_1 < \cdots < T_K = 1$.
2) Create a base RBM R_0 with parameters $\mathbf{\Theta}_0 = (\mathbf{W}_0, \mathbf{b}, \mathbf{c})$, where $\mathbf{W}_0 = 0$.
3) Create a sequence of intermediate RBMs R_k such that
   - It has twice as many hidden nodes as the target RBM has.
   - Parameters are $\mathbf{\Theta}_k = ([(1 - T_k)\mathbf{W}_0 \;\; T_k\mathbf{W}], \mathbf{b}, \left[\mathbf{c}^T \mathbf{c}^T\right]^T)$.
4) For m = 1 to M, repeat
   a) Sample $\mathbf{x}_1$ from R_0.
   b) For k = 1 to $K - 1$, repeat
      - Sample $\mathbf{x}_{k+1}$ from R_k by one-step Gibbs sampling starting from $\mathbf{x}_k$.
   c) 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.
5) The estimate of $\frac{Z_K}{Z_0}$ is $\frac{1}{M}\sum_{m=1}^{M} u_m$.

TABLE II: Sequence of estimating the normalizing constant by annealed importance sampling.

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 an 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 Table II.

To achieve an accurate estimate of the normalizing constant, a large number of intermediate RBMs should be used, and the normalizing constant must be estimated by as many annealing runs as possible [11]. This means that K and M in the algorithm from Table II should be large. The runs of AIS are independent from each other, so they can be fully parallelized.

VI. Experiments

Two different sets of experiments were done using MATLAB. The goal of the first experiment was to test the capability of RBMs to capture the data distribution. We generated samples from RBM trained on the OptDigits data set. The data set was acquired from the UCI Machine Learning Repository [12] 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 experiment was conducted in order to compare the performance of RBM depending on two different learning methods: CD learning and learning using sampling with PT. The performance was evaluated by the approximated likelihood of the training data set and the approximated probability of the test data set. We observed that the performance is better when the gradient was estimated using PT sampling.

Furthermore, in the second experiment we computed the probability of uniformly randomly generated data in 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 [13]. The probability of random data was computed for different learning methods and compared.[2]

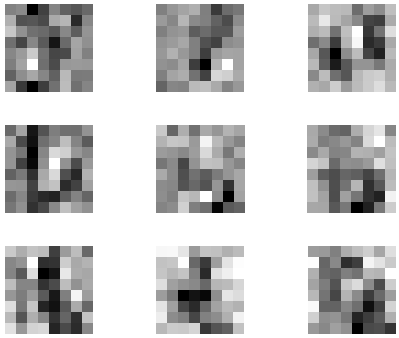
A. Generating samples from a trained restricted Boltzmann machine

RBM was constructed such that there are 64 visible neurons and 100 hidden neurons. Each neuron had a bias parameter. 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 we rounded the intensity of each pixel so that the intensity less than 8 became 0 (and 1 otherwise).

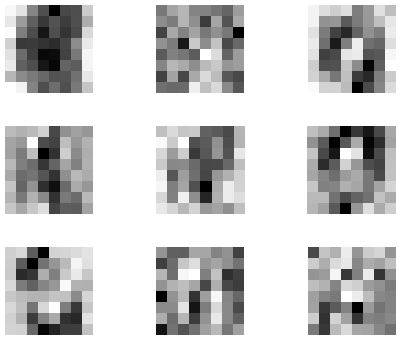
RBM was trained separately by CD learning with $n = 1$ and learning with PT sampling. PT sampling was done with $K = 20$ and temperatures $T_0 = 0$, $T_1 = 0.05, \ldots, T_{20} = 1$. The models represented by the RBMs are named CD1 and

[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)

Fig. 1: Training data set and visualization of hidden nodes. (a) shows 10 training samples where for each digit one sample was randomly chosen. (b) and (c) shows the weights connected to nine randomly chosen hidden neurons.

(a) Model learned with CD1

(b) Model learned with PT

Fig. 2: 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.

PT, respectively. Each gradient update was done in the 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 strategy.[3]

Figure 1 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 associated with the node as a grey-scale digit. It can be observed that each hidden node represents a distinct feature.

To see the generative behavior of RBM, the samples were gathered using PT sampling starting from a random initial sample. $K = 20$ was used for PT sampling. Figure 2 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 2 represents the random initial sample. It is clear that the trained RBM is able to generate digits which look similar to the training data. The proposed method of learning with PT sampling works as well as the conventional CD learning.

B. Comparison between contrastive divergence learning and parallel tempering

For the second experiment, we trained RBMs with the same 100 hidden neurons using four learning algorithms: CD1, CD5, CD25 and PT (where CDn is CD learning with n sampling steps).

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 training data samples. PT was, therefore, trained with $K = 20$ temperatures and $M = 192$ samples per gradient update. This choice is reasonable in a 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.

Each RBM was trained for 635 epochs and the probabilities of both training and test data were estimated. The parameters used in AIS were $M = 50$ and $K = 5000$. All the models were trained 30 times and the averaged performance indices were calculated.

Figure 3 shows that the probability of the test data increases, while the probability of the random data decreases over the gradient updates. This is consistent with the fact that the

[3] $\eta(n) = \frac{\eta_0}{1+\frac{n}{n_0}}$, where $\eta_0 = 0.05$ for both the weight and the bias, and $n_0 = 300$ for both CD1 and PT.

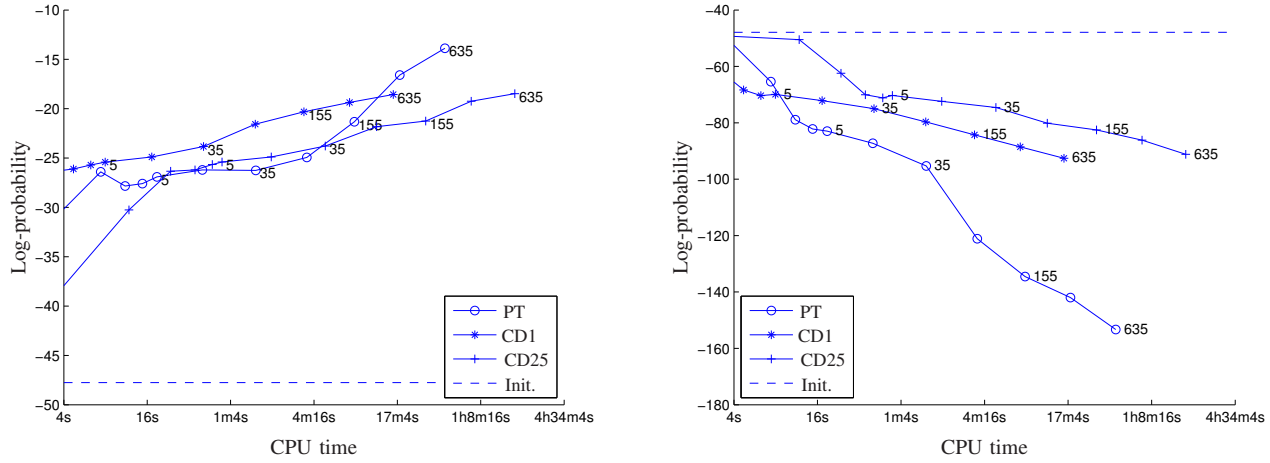


Fig. 3: Left: Average probabilities of test data against the processor time. Right: Average probabilities of random data against the processor time. The time scale is logarithmic. The dashed lines indicate the log-probability of the initial weights for both test and random data. The numbers denote the number of epochs after which the value was measured.

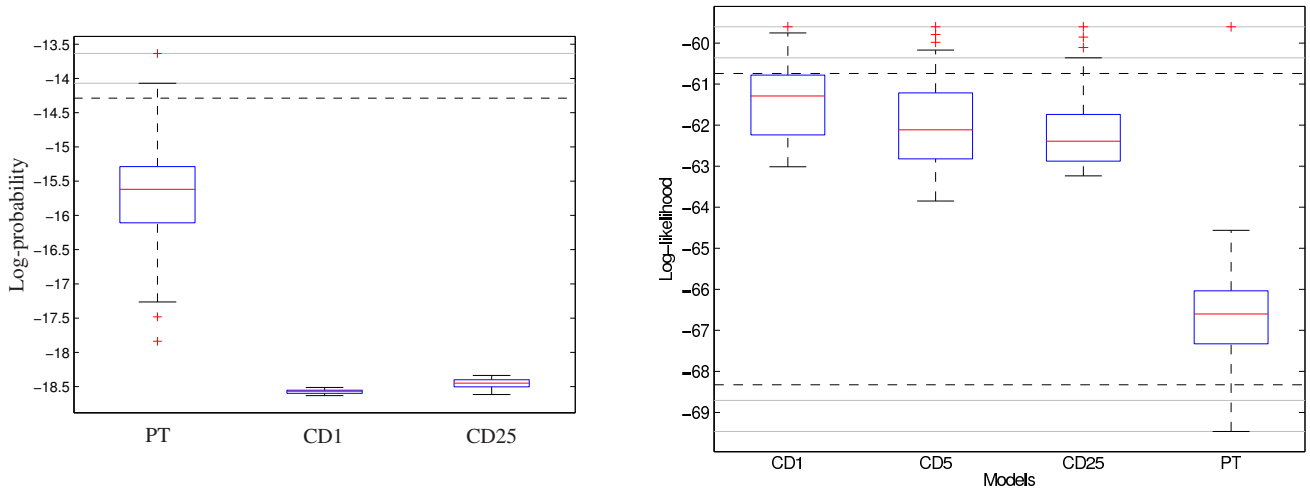


Fig. 4: Left: Box plots of probabilities of test data after 635 epochs over 30 repeated runs. Right: Box plots of probabilities of random data over 30 repeated runs. For probability values, the red line inside the box denotes the median, the edges of the box are 25-th and 75-th percentiles, and the whiskers are extended to the extreme value not considering possible outliers.

gradient maximizes the likelihood according to the distribution of the training data. It also 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 4 which shows the average probability of the test data set and the random data set by 30 independent trials, further, confirms 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.

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 as large as 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 RBM.

VII. Discussion

As an alternative to Gibbs sampling, contrastive divergence learning has been proposed and made learning of RBMs faster. Despite its computationally favorability and the wide acceptance, contrastive divergence learning is biased in the sense that the computed gradient computed does not lead

exactly to the maximum of the likelihood. This paper, therefore, proposed an alternative approach which utilizes parallel tempering for training RBMs. This approach does not sacrifice the optimality of the direction of the gradient but reduces the computational cost by improving the quality of the samples.

Two separate experiments were done: (1) to confirm the capability of RBM to capture the data distribution and (2) to show 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 log-likelihood estimated using annealed importance sampling 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 in [14]. The work in [14] illustrates the possible explanations why PT learning performs better than CD learning, and presents the experimental results showing the superiority of PT learning. It, however, lacks showing the efficiency of PT learning compared to CD learning in terms of the computational complexity, whereas we showed that even in the terms of the computational load PT learning is as efficient as CD learning.

A similar attempt at improving the learning by adapting an advanced sampling method has been proposed in [5]. The work also does sampling by considering multiple distributions of different temperatures, but the details of the proposed algorithm (called *tempered transition*) differ greatly from that presented in this paper.

Findings of this paper raise further research issues related to improving PT sampling for training RBM. The in-depth study of how the parameters of PT sampling influence the performance must be done, as the experiments in this report were done only with one specific setting of PT sampling. The adjustable parameters such as the learning rate, the number of temperatures and the number of samples, significantly affects the performance of learning, and the relationship between the choice of parameters and the performance must be further studied in order for PT sampling to be widely used.

Acknowledgements

This work was supported by the honours programme of the department, by the Academy of Finland and by the IST Program of the European Community, under the PASCAL2 Network of Excellence. This publication only reflects the authors' views.

REFERENCES

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

[2] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Artificial Intelligence and Statistics*, 2009.

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

[4] D. J. C. Mackay, *Information Theory, Inference & Learning Algorithms*, 1st ed. Cambridge University Press, June 2002.

[5] R. Salakhutdinov, "Learning in Markov random fields using tempered transitions," *NIPS 2009*, 2009.

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

[7] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comp.*, vol. 14, no. 8, pp. 1771–1800, August 2002.

[8] M. A. Carreira-Perpiñ and G. Hinton, "On contrastive divergence learning," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados*, R. G. Cowell and Z. Ghahramani, Eds. Society for Artificial Intelligence and Statistics, 2005, pp. 33–40.

[9] D. J. Earl and M. W. Deem, "Parallel tempering: Theory, applications, and new perspectives," *Phys. Chem. Chem. Phys.*, vol. 7, no. 23, pp. 3910–3916, 2005.

[10] R. M. Neal, "Annealed importance sampling," *Statistics and Computing*, vol. 11, pp. 125–139, 1998.

[11] R. Salakhutdinov, "Learning deep generative models," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[12] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[13] Y. Bengio, "Learning deep architectures for ai," Dept. IRO, Universite de Montreal, Tech. Rep., 2007.

[14] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau, "Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines," Dept. IRO, Universite de Montreal, Tech. Rep. 1345, 2009.