
Gaussian-Bernoulli Deep Boltzmann Machine

KyungHyun Cho, Tapani Raiko and Alexander Ilin

Aalto University School of Science

Department of Information and Computer Science

Espoo, Finland

{kyunghyun.cho, tapani.raiko, alexander.ilin}@aalto.fi

Abstract

In this paper, we study a model that we call Gaussian-Bernoulli deep Boltzmann machine (GDBM) and discuss potential improvements in training the model. GDBM is designed to be applicable to continuous data and it is constructed from Gaussian-Bernoulli restricted Boltzmann machine (GRBM) by adding multiple layers of binary hidden neurons. The studied improvements of the learning algorithm for GDBM include parallel tempering, enhanced gradient, adaptive learning rate and layer-wise pretraining. We empirically show that they help avoid some of the common difficulties found in training deep Boltzmann machines such as divergence of learning, the difficulty in choosing right learning rate scheduling, and the existence of meaningless higher layers.

1 Introduction

Deep Boltzmann machine (DBM) [20] is a recent extension of the simple restricted Boltzmann machine (RBM) in which several RBMs are stacked on top of each other. Unlike in other models, such as deep belief network or deep autoencoders (see, e.g., [11, 1]), in DBM, each neuron in the intermediate layers receives both top-down and bottom-up signals, which facilitates propagation of uncertainty during the inference procedure [20]. The original DBM is constructed such that each visible neuron represents a binary variable, that is DBM learns distributions over binary vectors.

A popular approach to modeling real-valued data is normalizing each input variable to $[0, 1]$ and treating it as a probability (e.g., using a gray-scale value of a pixel as a probability [11, 14]). This approach is however restrictive and it fits best to bounded variables. In the original DBM [20], real-valued data are first transformed into binary codes by training a model called Gaussian-Bernoulli RBM (GRBM) [11], and DBM is learned for the binary codes extracted from the data. This approach showed promising results [20, 19, 21] but it may be beneficial to combine GRBM and DBM in a single model and allow their joint optimization.

In this paper, we study a Gaussian-Bernoulli deep Boltzmann machine (GDBM) which uses Gaussian units in the visible layer of DBM. Even though deriving stochastic gradient is rather easy for GDBM, the training procedure can easily run into problems without careful selection of the learning parameters. This is largely caused by the fact that GRBM is known to be difficult to tune, especially the variance parameters of the visible neurons (see, e.g., [12]). We propose an algorithm for training GDBM based on the improvements introduced recently for training GRBM [3]. Also, we discuss the universal approximator property of GDBM.

The rest of the paper is organized as follows. The GDBM model is introduced in Section 2. In Section 3, we present the training algorithm and explain how to compute the terms of the stochastic gradient using mean-field approximations and parallel tempering sampling. In Section 4.1, we describe the update rules that are invariant to the data representation and more robust to the initialization of the parameters. In Section 4.2, we describe how we adapt the learning rate using the ideas proposed in [5].

2 Gaussian-Bernoulli Deep Boltzmann Machine

GDBM with a single visible layer and L hidden layers is parameterized with weights \mathbf{W} of synaptic connections between the visible layer and the first hidden layer, $\mathbf{W}^{(l)}$ between layers l and $l + 1$, biases \mathbf{b} of the visible layer, $\mathbf{b}^{(l)}$ of each hidden layer l , and standard-deviations σ_i of the visible neurons. For a certain state $[\mathbf{v}^T \mathbf{h}^{(1)T} \dots \mathbf{h}^{(L)T}]^T$, the energy is defined as:

$$E(\mathbf{v}, \mathbf{h}^{(1)}, \dots, \mathbf{h}^{(L)} | \boldsymbol{\theta}) = \sum_{i=1}^{N_v} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^{N_v} \sum_{j=1}^{N_1} \frac{v_i}{\sigma_i^2} h_j^{(1)} w_{ij} \\ - \sum_{l=1}^L \sum_{j=1}^{N_l} b_j^{(l)} h_j^{(l)} - \sum_{l=1}^{L-1} \sum_{j=1}^{N_l} \sum_{k=1}^{N_{l+1}} h_j^{(l)} h_k^{(l+1)} w_{jk}^{(l)}$$

and the corresponding probability is

$$p(\mathbf{v}, \mathbf{h}^{(1)}, \dots, \mathbf{h}^{(L)} | \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\mathbf{v}, \mathbf{h}^{(1)}, \dots, \mathbf{h}^{(L)} | \boldsymbol{\theta})\right), \quad (1)$$

where N_v and N_l are the number of neurons in the visible layer and the l -th hidden layer, respectively, and $Z(\boldsymbol{\theta})$ is the normalizing constant. Note that we use the GRBM parameterization, including learning $z_i = \log \sigma_i^2$ instead of σ_i directly, from [3]. Note also that GRBM is a special case of GDBM with $L = 1$.

The states of the neurons in the same layer are independent of each other given the adjacent upper and lower layers. The conditional probability of a visible neuron is

$$p(v_i | \mathbf{h}^{(1)}, \boldsymbol{\theta}) = \mathcal{N}\left(v_i \mid \sum_{j=1}^{N_1} h_j^{(1)} w_{ij} + b_i, \sigma_i^2\right),$$

where $\mathcal{N}(\cdot | \mu, \sigma^2)$ is a probability density of Normal distribution with a mean μ and a standard deviation σ , and the conditional probabilities of the hidden neurons are

$$P(h_j^{(1)} | \mathbf{v}, \mathbf{h}^{(2)}, \boldsymbol{\theta}) = f\left(\sum_{i=1}^{N_v} \frac{v_i}{\sigma_i^2} w_{ij} + \sum_{k=1}^{N_2} h_k^{(2)} w_{jk}^{(1)} + b_j^{(1)}\right), \\ P(h_j^{(l)} | \mathbf{h}^{(l-1)}, \mathbf{h}^{(l+1)}, \boldsymbol{\theta}) = f\left(\sum_{i=1}^{N_{l-1}} h_i^{(l-1)} w_{ij}^{(l-1)} + \sum_{k=1}^{N_{l+1}} h_k^{(l+1)} w_{jk}^{(l)} + b_j^{(l)}\right),$$

where $f(\cdot)$ is a sigmoid function and $N_{L+1} = 0$.

2.1 GDBM is a universal approximator

GRBM belongs to the family of mixture of Gaussians (MoG) since its joint distribution can be factorized into $p(\mathbf{v}, \mathbf{h}^{(1)}) = P(\mathbf{h}^{(1)})p(\mathbf{v} | \mathbf{h}^{(1)})$ where $P(\mathbf{h}^{(1)})$ is the mixture coefficients and $p(\mathbf{v} | \mathbf{h}^{(1)})$ is Gaussian. MoGs are known to be universal approximators [8]. However, the center points of the exponentially many Gaussians in the data space are defined by only a linear number of parameters w.r.t. the number of hidden units, so not all MoGs can be written as a GRBM.

Given a MoG, we could transform it into a GRBM if we further constrain that exactly one of the hidden units is active at a time, that is, $\sum_j h_j^{(1)} = 1$. We set the columns of \mathbf{W} to match the center points of the MoG and \mathbf{b} to $\mathbf{0}$, and $\mathbf{b}^{(1)}$ is set such that the marginals $P(\mathbf{h}^{(1)})$ would match the mixing coefficients of the MoG.

As the final step, we implement the restriction $\sum_j h_j^{(1)} = 1$ using another hidden layer. We set $N_2 = N_1$ and $b_j^{(2)} = -\omega$ for each j , $w_{ij}^{(1)} = 3\omega$ for all $i = j$ and $w_{ij}^{(1)} = -\omega$ for all $i \neq j$, and further subtract ω from each $b_i^{(1)}$. As ω goes to infinity, it is clear that the probability of all states where $\mathbf{h}^{(2)} \neq \mathbf{h}^{(1)}$ or $\sum_j h_j^{(1)} \neq 1$ goes to zero and other states implement the previous GRBM with the wanted restriction. We have thus shown that any MoG can be modeled with a GDBM, and GDBM is a universal approximator.

3 Training GDBM

GDBM can be trained with the stochastic maximization of the likelihood, where the likelihood function is computed by marginalizing out all the hidden neurons. For each parameter θ , the partial-derivative of the log-likelihood function is

$$\frac{\partial \mathcal{L}}{\partial \theta} \propto \left\langle \frac{\partial (-E(\mathbf{v}^{(t)}, \mathbf{h} | \boldsymbol{\theta}))}{\partial \theta} \right\rangle_{\mathbf{d}} - \left\langle \frac{\partial (-E(\mathbf{v}, \mathbf{h} | \boldsymbol{\theta}))}{\partial \theta} \right\rangle_{\mathbf{m}}, \quad (2)$$

where $\langle \cdot \rangle_{\mathbf{d}}$ and $\langle \cdot \rangle_{\mathbf{m}}$ denote the expectation over the data distribution $P(\mathbf{h} | \{\mathbf{v}^{(t)}\}, \boldsymbol{\theta})$ and the model distribution $P(\mathbf{v}, \mathbf{h} | \boldsymbol{\theta})$, respectively. $\{\mathbf{v}^{(t)}\}$ is a set of all the training samples.

3.1 Computing expectation over the data distribution

Computing the first term of (2) is straightforward for restricted Boltzmann machines because in that model the hidden neurons are independent of each other given the visible neurons. However, this does not apply to GDBM and therefore one needs to use some sort of approximation. We employ the mean-field approximation that was used for training binary DBM in [20].

In the mean-field approximation, the visible neurons are fixed to a training data sample, and the state of each hidden neuron $h_j^{(l)}$ is described with its probability $\mu_j^{(l)}$ of being active, which is updated with the following fixed-point iterations:

$$\mu_j^{(l)} \leftarrow f \left(\sum_{i=1}^{N_{l-1}} \mu_i^{(l-1)} w_{ij}^{(l-1)} + \sum_{k=1}^{N_{l+1}} \mu_k^{(l+1)} w_{jk}^{(l)} + b_j^{(l)} \right),$$

Note that $N_0 = N_v$, $\mu_i^{(0)} = v_i / \sigma_i^2$, and the update rule for the top-most layer not contain the summation term $\sum_{k=1}^{N_{l+1}}$. Using the mean-field approximation is known to introduce a bias. However, this is a computationally efficient scheme to capture only one mode of the posterior distribution, which can be desirable in many practical applications [20].

3.2 Computing expectation over the model distribution: Parallel Tempering approach

The second term of the gradient (2) can be computed using Markov-chain Monte-Carlo (MCMC) sampling. The original approach proposed in [18] uses persistent Gibbs sampling with only a few steps of sampling at each update. This is equivalent to persistent contrastive divergence (PCD) introduced for training RBM [24]. Unfortunately the persistent Gibbs sampling suffers from poor mixing of samples, which results in the fact that trained models may have probability mass in the areas which are not represented in the training data (false modes) [7, 18, 19, 2]. In our experiments, we were able to observe that learning can easily diverge when persistent Gibbs sampling is used (see Section 5).

We therefore use parallel tempering recently proposed in the context of RBM and GRBM [7, 4, 3] as a sampling procedure for GDBM. Parallel tempering overcomes the poor-mixing problem by maintaining multiple chains of sampling with different temperatures. In the chain with a high temperature, particles are more likely to explore the state space easily, whereas particles in the chain with low temperatures more closely follow the target model distribution.

We define the tempered distributions by varying parameters $\boldsymbol{\theta}$ of the original GDBM (1). We denote by $\boldsymbol{\theta}_\beta$ the parameters of the intermediate models defined by *inverse* temperatures β , where $\beta = 0$ corresponds to the base (most diffuse) distribution and $\beta = 1$ corresponds to the target distribution defined by the original GDBM. Defining appropriate intermediate distributions is quite straightforward for binary RBM [7, 4] but it is not as trivial for models with real-valued visible units [3]. In this work, we use the tempering scheme defined by the following choice of $\boldsymbol{\theta}_\beta$:

$$\begin{aligned} \mathbf{b}_\beta &= \beta \mathbf{b} + (1 - \beta) \mathbf{m}, & \mathbf{b}_\beta^{(l)} &= \beta \mathbf{b}^{(l)} \quad (l \geq 1), \\ \sigma_{\beta,i} &= \sqrt{\beta \sigma_i^2 + (1 - \beta) s_i^2}, & \mathbf{W}_\beta^{(l)} &= \beta \mathbf{W}^{(l)}, \end{aligned}$$

where $\mathbf{m} = [m_i]_{i=1}^{N_v}$ and s_i are the means and variances estimated from the samples obtained from *all the tempered distributions*. Thus, the base distribution is the Gaussian distribution fitted to the samples from all the intermediate chains. The proposed scheme assures that the swapping happens even if the target distribution diverges from the data distribution. According to our experiments, this results in more stable learning compared to the scheme proposed in [3].

Adapting the temperature during learning can improve mixing and therefore facilitate learning [6]. In our experiments, we adapt the temperatures so as to maintain that the numbers of particle swaps between the consecutive chains as equal as possible. This is done by adjusting the inverse temperatures $\{\beta_i, i = 1, \dots, N_{\text{chains}}\}$ after every swapping round using the following heuristic:

$$\beta_i^{(t)} \leftarrow \eta \beta_i^{(t-1)} + (1 - \eta) \frac{\sum_{j=1}^{i-1} n_j}{\sum_{k=1}^{N_{\text{chains}}-1} n_k},$$

where η is the damping factor, n_j denote the number of swaps between the chains defined by β_j and β_{j+1} , and the hottest chain is kept at the initial temperature, that is $\beta_1 = 0$. This simple approach does not have much computational overhead and seems to improve learning, as we show in Section 5.

The proposed scheme of adapting the base distribution and the temperatures seem to work well in practice, although it may introduce a bias. The analysis of the possible biases of this approach is a question for further research.

4 Improving the Training Procedure

In this section, we show how to improve training of GDBM by adapting several ideas introduced for training RBM and deep networks.

4.1 Enhanced gradient for GDBM

Enhanced gradient was introduced recently to make the update rules of binary Boltzmann machines invariant to data representation [17, 5]. The gradient was derived by introducing bit-flipping transformations and making the update rule, which follows from (2), invariant to such transformations. It has been shown to improve learning of RBM by making the results less sensitive to the learning parameters and initialization.

The same ideas can be used for enhancing the gradient in the models with Gaussian visible neurons such as GRBM and GDBM. Instead of the bit-flipping transformations, one can transform the original model by shifting each visible unit as $\tilde{v}_i = v_i - \Delta_i$ and correcting the bias terms accordingly: $\tilde{b}_i = b_i + \Delta_i$ and $\tilde{b}_i^{(1)} = b_i^{(1)} - \sum_i^N \frac{\Delta_i}{\sigma_i^2} w_{ij}$, which would result in an equivalent model. Following the methodology of [5], one can select the shifting parameters Δ_i such that the resulting gradient w.r.t. weights $w_{ij}^{(l)}$ and biases $b_i^{(l)}$ do not contain the same terms. This yields the following update rules:

$$\begin{aligned} \nabla_e w_{ij} &= \text{Cov}_d \left(\frac{v_i}{\sigma_i^2}, h_j^{(1)} \right) - \text{Cov}_m \left(\frac{v_i}{\sigma_i^2}, h_j^{(1)} \right), \\ \nabla_e w_{ij}^{(l)} &= \text{Cov}_d \left(h_i^{(l)}, h_j^{(l+1)} \right) - \text{Cov}_m \left(h_i^{(l)}, h_j^{(l+1)} \right) \quad (l = 1, \dots, L-1), \\ \nabla_e b_i &= \nabla b_i - \sum_j \left\langle h_j^{(1)} \right\rangle_{\text{dm}} \nabla_e w_{ij}, \\ \nabla_e b_i^{(1)} &= \nabla b_i^{(1)} - \sum_j \left\langle h_j^{(2)} \right\rangle_{\text{dm}} \nabla_e w_{ij}^{(1)} - \sum_k \left\langle v_k \right\rangle_{\text{dm}} \nabla_e w_{ki}, \\ \nabla_e b_i^{(l)} &= \nabla b_i^{(l)} - \sum_j \left\langle h_j^{(l+1)} \right\rangle_{\text{dm}} \nabla_e w_{ij}^{(l)} - \sum_k \left\langle h_k^{(l-1)} \right\rangle_{\text{dm}} \nabla_e w_{ki}^{(l-1)} \quad (l > 1), \end{aligned}$$

where $\left\langle h_j^{(l)} \right\rangle_{\text{dm}} = \frac{1}{2} \left\langle h_j^{(l)} \right\rangle_d + \frac{1}{2} \left\langle h_j^{(l)} \right\rangle_m$ is the average activity of a neuron under the data and model distributions and $\left\langle v_i \right\rangle_{\text{dm}} = \frac{1}{2} \left\langle v_i / \sigma_i^2 \right\rangle_d + \frac{1}{2} \left\langle v_i / \sigma_i^2 \right\rangle_m$. $\text{Cov}_P(\cdot, \cdot)$ is a covariance between

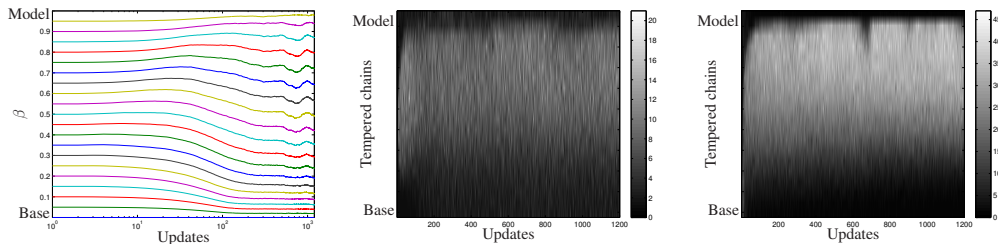


Figure 1: The left figure shows the evolution of the inverse temperatures during the learning while β_1 is fixed to 0. The middle and right figures plot the number of swaps between a pair of consecutive samples at each update (swap) while the temperatures were adapted (middle) and the temperatures were fixed at the equally spaced levels (right).

two variables under the distribution P defined as

$$\text{COV}_P(v_i, h_j) = \langle v_i h_j \rangle_P - \langle v_i \rangle_P \langle h_j \rangle_P.$$

4.2 Adaptive learning rate

The choice of the learning rate to be used with the stochastic gradient (2) greatly affects the training procedure [10, 23, 5]. In order to diminish this problem, we adopt the strategy of automatic adaptation of the learning rate, as proposed in [5]. The adaptation is done based on the estimate of the likelihood computed using the identity

$$p(\mathbf{v}_d | \boldsymbol{\theta}_\eta) = \frac{p^*(\mathbf{v}_d | \boldsymbol{\theta}_\eta)}{Z(\boldsymbol{\theta})} \left\langle \frac{p^*(\mathbf{v} | \boldsymbol{\theta}_\eta)}{p^*(\mathbf{v} | \boldsymbol{\theta})} \right\rangle_{p(\mathbf{v} | \boldsymbol{\theta})}^{-1}, \quad (3)$$

where $\boldsymbol{\theta}_\eta$ are the model parameters obtained by updating $\boldsymbol{\theta}$ with learning rate η , p^* denotes an unnormalized pdf such that $p(\mathbf{v} | \boldsymbol{\theta}) = p^*(\mathbf{v} | \boldsymbol{\theta}) / Z(\boldsymbol{\theta})$ and the required expectation is approximated using samples from $p(\mathbf{v} | \boldsymbol{\theta})$.

The unnormalized probabilities $p^*(\mathbf{v} | \boldsymbol{\theta})$ are obtained by integrating out the hidden neurons from the joint model:

$$p^*(\mathbf{v} | \boldsymbol{\theta}) = \sum_{\mathbf{h}} p^*(\mathbf{v}, \mathbf{h}),$$

which yields a simple analytical form in the case of RBM or GRBM. However, explicit integration of the hidden neurons is not tractable for GDBM and therefore one has to employ some approximations. We use the approximation

$$\sum_{\mathbf{h}} E(\mathbf{v}, \mathbf{h}) \approx E(\mathbf{v}, \boldsymbol{\mu}),$$

where $\boldsymbol{\mu}$ is the mean-field approximation of the hidden activations which is computed as discussed in Section 3.1.

Note that we use different portions of data (mini-batches) for computing the stochastic gradient and adaptation of the learning rate, as was suggested in [5]. Therefore, the fixed-point iterations required for computing the mean-field approximation have to be run twice. However, initializing the mean-field values with samples from the model distribution seems to yield fast convergence. Also, making only a few fixed-point iterations (without convergence) seems to be enough to get stable behavior of the learning rate adaptation.

4.3 Layer-wise pretraining

Layer-wise pretraining is commonly used in deep networks to help obtain better models by initializing weights sensibly [9]. DBM requires special care during the pretraining phase because the neurons in the intermediate layers receive signal both from the upper and the lower layers, unlike

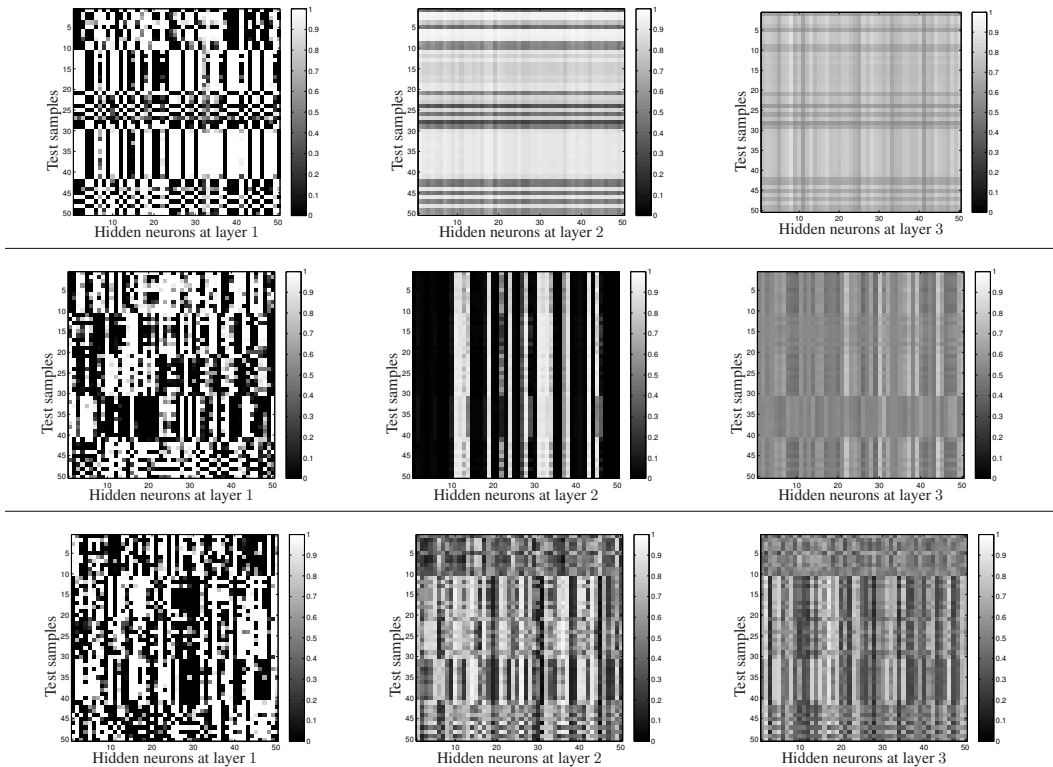


Figure 2: The figures visualize the mean-field values of the hidden neurons at different hidden layers when the visible neurons are fixed to the test data samples. The top figures were obtained using the GDBM trained without any pretraining. The middle and bottom figures were obtained from the GDBM trained with both the pretraining and the joint-training using either using the traditional gradient or the enhanced gradient, respectively. All three GDBMs were trained for 215 epochs.

in deep belief networks [11]. Salakhutdinov proposed cope with this problem by halving the pre-trained weights in the intermediate layers and duplicating the visible and topmost layers during the pretraining [18]. The pre-trained GRBM containing the visible layer has the following energy:

$$E(\mathbf{v}, \mathbf{h}^{(1)} | \boldsymbol{\theta}) = \sum_i \frac{(v_i - b_i)^2}{2\sigma_i^2/N_v} - \sum_j c_j h_j^{(1)} - \sum_i \sum_j \frac{v_i}{\sigma_i^2 N_v} h_j^{(1)} w_{ij},$$

where $N_v = 2$ corresponds to duplicating the visible layer. Similarly, the topmost RBM during pretraining has the energy

$$E(\mathbf{h}^{(L-1)}, \mathbf{h}^{(L)} | \boldsymbol{\theta}) = - \sum_i b_i h_i^{(L-1)} - \sum_j (N_h c_j) h_j^{(L)} - \sum_i \sum_j h_i^{(L-1)} h_j^{(L)} (N_h w_{ij}^{(L-1)}),$$

where we also use $N_h = 2$.

5 Experiments

We train the GDBM model on Olivetti face dataset [22]. Out of 400 faces, we used the first 350 faces of 35 people for training and the remaining ones of 5 people as test samples. We test the model in the problem of reconstructing the left half of the face from the right half. We compared the reconstruction results using the methodology presented in [16]. Note that the test set does not contain faces of people from the training set for better assessment of the generalization skills. The dataset was initially normalized such that each pixel has zero-mean and unit variance. We trained GDBM models with three hidden layers and 500 neurons in each hidden layer.

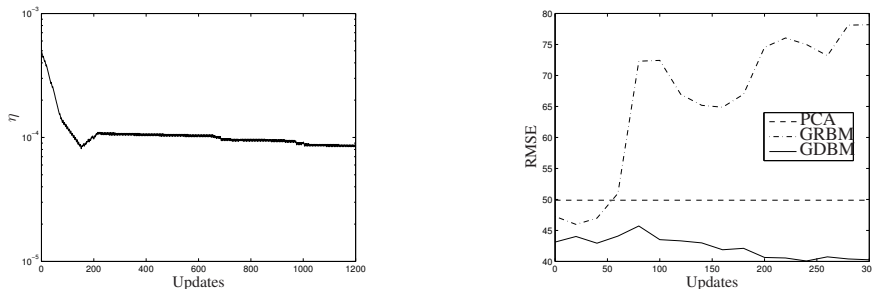


Figure 3: The evolution of the learning rate when the adaptive learning rate was used (left) and the reconstruction errors using three different methods: PCA, GRBM, and GDBM (right).

Unless specified otherwise, GDBM is trained using pretraining, enhanced gradient and adaptive learning rate. The size of a minibatch and the number of samples from the model distribution were both set to 64. The initial learning rate and its upper-bound were both set to 0.001 for pretraining and to 0.0005 for joint training of all the layers. Weight-decay of 0.005 was used both during training RBM and pretraining DBM. The reconstruction was performed by fixing the known half of the image, computing the mean-field approximation of the posterior distribution over all the unknown neurons including the missing half of the image.

Adaptive Learning Rate. Our experiments confirmed that the learning rate was able to adapt automatically using the proposed strategy. The left plot in Figure 3 shows that the algorithm very quickly finds the appropriate region of learning rates. After that, the learning rate slowly decreases, which is a desired property in stochastic optimization (see, e.g., [13]).

Parallel Tempering. Parallel tempering yields pretty good results (see Fig. 4) while we were not able to achieve convergence of GDBM with PCD. Fig. 1 indicates that proposed scheme of temperature adaptation results in the increased and consistent number of swaps among all consecutive pairs of tempered chains, and hence, in better mixing.

Enhanced Gradient and Pretraining. One obvious way to check whether the higher layers of GDBM have learned any useful structure is to inspect the mean-field approximation values of the neurons in those layers given the training or test data samples. When no useful structure was learned, most hidden neurons in the top layer converge near 0.5 which means that nearly no signal was received from the neighboring layers. On the other hands, when those neurons actually affect the modeling of the distribution, they converge to the values close to either 0 or 1.

One GDBM was trained without the layer-wise pretraining, starting from the random initialization. In this case it was clear that except for the first hidden layer no upper layer was able to learn any useful structure, as shown on the top row of Figure 2. Most of approximated values of the hidden neurons in the second and third layers are near 0.5.

We trained the second GDBM by first initializing it with layer-wise pretraining. However, this time, we did not use the enhanced gradient for either the pretraining or the joint training. Comparing with the top row of Figure 2, it is apparent from the figures of the middle row that the hidden neurons in the upper layers are approximated closer to 0 or 1 when the layers were pretrained. It suggests that the pretraining is important in a sense that it enables the upper layers to learn the structure of the data distribution.

However, we were able to observe that many hidden neurons in the higher layers (see the hidden layer 2, for instance) do not contribute much to the modeling of the distribution, as they are either always inactive ($= 0$) or always active ($= 1$). This behavior was already discovered in case of RBM, and it was shown that the enhanced gradient can resolve the problem [5].

Thus, we trained yet another GDBM now by using the enhanced gradient. The bottom row of Figure 2 clearly indicates that the enhanced gradient was able to address the problem. Now the hidden neurons in the layer 3 respond differently to the distinct test samples, and by doing so, encourages the flow of the uncertainty between the hidden layer 1 and the hidden layer 3, enabling the hidden neurons in the layer 3 to capture the structure also.

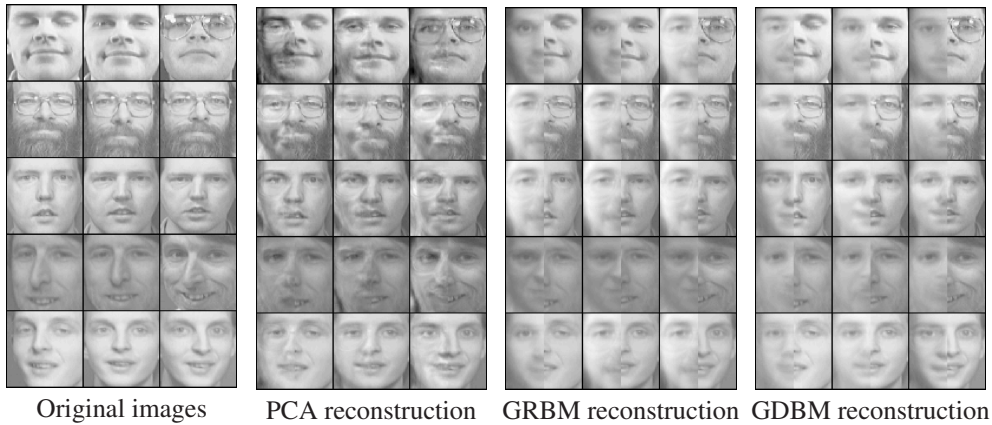


Figure 4: Reconstructed test samples using PCA, GRBM and GDBM.

Comparison with other models. We trained principal component analysis (PCA) and GRBM on the same dataset. PCA used 100 principal components [16], and GRBM had 500 hidden neurons. We limited the number of the principal components to 100 because including more components resulted in stronger overfitting and larger reconstruction errors.

The right plot in Fig. 3 shows the evolution of the difference between the original test faces and the reconstructed faces for each model. It indicates that GRBM start overfitting quickly, which results in the increase of the reconstruction error of the unseen faces. It is also evident from the reconstructed faces in Fig. 4. Ultimately, GRBM performs worse than PCA evidenced by both RMSE and the visual inspection of the reconstructed faces.

GDBM trained only with pretraining using the identical GRBM training procedure, however, does not become overfitted and performs better than GRBM, which indicates that the additional hidden layers help perform and generalize better.

Furthermore, the joint training of GDBM by the proposed learning algorithm improves the performance significantly. It suggests that it is indeed important to jointly train all layers of GDBM in order to obtain a better generative model.

6 Future Work and Discussion

In this paper, we defined Gaussian-Bernoulli deep Boltzmann machine and discussed its universal approximator property. Based on the learning algorithm for the binary DBM [20], we adapted three improvements which are parallel tempering, enhanced gradient, and adaptive learning rate for training a GDBM. Through the experiments we were able to empirically show that GDBM trained using these improvements can achieve good generative performance.

Although they are not presented in this paper, using the same hyper-parameters for learning faces, we were able to train GDBM on other data sets such as NORB [15] and CIFAR-10 [12]. It clearly indicates that the proposed improvements make learning insensitive to the choice of the learning hyper-parameters and thus easier. However, the trained GDBMs were not able to produce any state-of-the-art classification accuracy. The discrepancy between the generative capability and the classification performance of GDBM is left for the future research.

Recently, several approaches have been proposed for efficiently training DBM, and it will be interesting to see how they perform when they are used for training GDBM compared to the learning algorithm proposed in this paper. Some of them are; adaptive MCMC sampling [19], tempered-transition [18], and using a separate set of recognition weights [21].

References

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

- [2] K. Cho. Improved Learning Algorithms for Restricted Boltzmann Machines. Master's thesis, Aalto University School of Science, 2011.
- [3] K. Cho, A. Ilin, and T. Raiko. Improved Learning of Gaussian-Bernoulli Restricted Boltzmann Machines. In *Proceedings of the Twentieth International Conference on Artificial Neural Networks*, ICANN 2011, 2011.
- [4] K. Cho, T. Raiko, and A. Ilin. Parallel tempering is efficient for learning restricted boltzmann machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)*, Barcelona, Spain, July 2010.
- [5] K. Cho, T. Raiko, and A. Ilin. Enhanced Gradient and Adaptive Learning Rate for Training Restricted Boltzmann Machines. In *Proceedings of the Twenty-seventh International Conference on Machine Learning*, ICML 2011, 2011.
- [6] G. Desjardins, A. Courville, and Y. Bengio. Adaptive Parallel Tempering for Stochastic Maximum Likelihood Learning of RBMs. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.
- [7] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau. Parallel Tempering for Training of Restricted Boltzmann Machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 145–152, 2010.
- [8] A. S. D.M. Titterton and U. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons, New York, London, Sydney, 1985.
- [9] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 2010.
- [10] A. Fischer and C. Igel. 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, pages 208–217, Berlin, Heidelberg, 2010. Springer-Verlag.
- [11] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.
- [12] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto, 2009.
- [13] H. Kushner and G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2003.
- [14] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Meeting of the Cognitive Science Society*, 2011.
- [15] Y. Lecun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of CVPR'04*, volume 2, 2004.
- [16] H. Poon and P. Domingos. Sum-Product networks: A new deep architecture. In *Proceedings of the Twenty Seventh Conference on Uncertainty in Artificial Intelligence*, 2011.
- [17] T. Raiko, K. Cho, and A. Ilin. Enhanced gradient for learning boltzmann machines (abstract). In *The Learning Workshop*, Fort Lauderdale, Florida, April 2011.
- [18] R. Salakhutdinov. Learning in Markov Random Fields using Tempered Transitions. In Bengio, Y. and Schuurmans, D. and Lafferty, J. and Williams, C. K. I. and Culotta, A., editor, *Advances in Neural Information Processing Systems 22*, pages 1598–1606, 2009.
- [19] R. Salakhutdinov. Learning Deep Boltzmann Machines using Adaptive MCMC. In J. Frnkranz and T. Joachims, editors, *ICML*, pages 943–950. Omnipress, 2010.
- [20] R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.
- [21] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. *Journal of Machine Learning Research - Proceedings Track*, 9:693–700, 2010.
- [22] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision, 1994.*, pages 138 –142, dec 1994.
- [23] H. Schulz, A. Müller, and S. Behnke. Investigating Convergence of Restricted Boltzmann Machine Learning. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.
- [24] T. Tieleman. *Training restricted Boltzmann machines using approximations to the likelihood gradient*. ICML '08. ACM, New York, NY, USA, 2008.