A Two-Stage Pretraining Algorithm for Deep Boltzmann Machines

Kyunghyun Cho, Tapani Raiko, Alexander Ilin, and Juha Karhunen

1 Introduction

Deep Boltzmann machine (DBM), proposed in [33], is a recently introduced variant of Boltzmann machines which extends a widely used restricted Boltzmann machine (RBM) to have multiple layers of hidden neurons. It differs from the popular deep belief network (DBN) which is built by stacking multiple layers of RBMs [19] in that every edge in the DBM model is undirected. This construction of DBMs facilitates propagating uncertainties across multiple layers of hidden variables.

Although it is straightforward to derive a learning algorithm for DBMs using a variational approximation and stochastic maximum likelihood method, recent research (see, for example, [33] and [13]) has shown that learning the parameters of DBMs is not trivial. Especially the generative performance of the trained model, commonly measured by the variational lower-bound of log-probabilities of test samples, tends to degrade as more hidden layers are added.

In [33] a greedy layer-wise pretraining algorithm was proposed to initialize the parameters of DBMs. It was shown that the proposed algorithm largely overcomes the difficulty of learning a good generative model.

Along this line of research, we propose another strategy of pretraining DBMs in this paper. The proposed scheme is based on an observation that training DBMs consists of two separate stages; approximating the posterior distribution over hidden units and updating parameters to maximize the lower-bound of the log-likelihood given the approximate posterior distribution.

Based on this observation, the proposed algorithm in this paper pretrains a DBM in two stages. During the first stage we train a simpler, directed deep model such as DBNs or stacked denoising autoencoders (sDAE) to obtain an approximate posterior distribution over hidden units. With this approximate posterior distribution fixed,

Kyunghyun Cho · Tapani Raiko · Alexander Ilin · Juha Karhunen
Department of Information and Computer Science, Aalto University School of Science, Finland
e-mail: firstname.lastname@aalto.fi

we train an RBM that learns a distribution over a combination of data samples and their corresponding posterior distributions over the hidden units. It is then trivial to finetune the model as one only needs to simply free hidden variables from the fixed approximate posterior distribution obtained in the first stage.

We show that the proposed algorithm helps learning a good generative model which is empirically comparable to, or better than the pretraining method proposed in [33]. Furthermore, we discuss the degrees of freedom in extending the proposed approach.

2 Deep Boltzmann Machines

We start by describing deep Boltzmann machines (DBM) [33]. A DBM with L layers of hidden neurons is defined by the following negative energy function:

$$-E(\mathbf{v},\mathbf{h} \mid \theta) = \sum_i^{N_v} v_i b_i + \sum_i^{N_v}\sum_j^{N_1} v_i h_j^{[1]} w_{i,j} + \sum_j^{N_1} h_j^{[1]} c_j^{[1]} + \sum_{l=2}^{L}\left(\sum_j^{N_l} h_j^{[l]} c_j^{[l]} + \sum_j^{N_l}\sum_k^{N_{l+1}} h_j^{[l]} h_k^{[l+1]} u_{j,k}^{[l]}\right), \quad (1)$$

where $\mathbf{v} = [v_i]_{i=1\ldots N_v}$ and $\mathbf{h}^{[l]} = \left[h_j^{[l]}\right]_{j=1\ldots N_l}$ are N_v binary visible units and N_l binary hidden units in the l-th hidden layer, where $1 \leq l \leq L$.

$\mathbf{W} = [w_{i,j}]_{i=1\ldots N_v, j=1\ldots N_1}$ is the set of weights between the visible neurons and the first layer hidden neurons. $\mathbf{U}^{[l]} = \left[u_{j,k}^{[l]}\right]_{i=1\ldots N_l, j=1\ldots N_{l+1}}$ is the set of weights between the l-th and $(l+1)$-th hidden neurons, where $1 \leq l < L$, in this case. b_i and $c_j^{[l]}$ are a bias to the i-th visible neuron and the j-th hidden neuron in the l-th hidden layer, respectively. We use θ to denote a set of all these parameters.

With the energy function, a DBM can assign a probability to each state vector

$$\mathbf{x} = [\mathbf{v}; \mathbf{h}^{[1]}; \cdots; \mathbf{h}^{[L]}]$$

using a Boltzmann distribution

$$p(\mathbf{x} \mid \theta) = \frac{1}{Z(\theta)} \exp\{-E(\mathbf{x} \mid \theta)\},$$

where

$$Z(\theta) = \sum_{\forall \mathbf{x}} \exp\{-E(\mathbf{x} \mid \theta)\}.$$

Under this formulation, the conditional distribution of each hidden layer l is

$$p(h_j^{[l]} = 1 \mid \mathbf{h}^{[l-1]}, \mathbf{h}^{[l+1]}, \theta) = f\left(\sum_{k=1}^{q_{l-1}} h_k^{[l-1]} u_{kj}^{[l-1]} + \sum_{i=1}^{q_{l+1}} h_i^{[l+1]} u_{ji}^{[l]} + c_j^{[l]}\right), \quad (2)$$

and the conditional distribution of the visible layer is

$$p(v_i = 1 \mid \mathbf{h}^{[1]}, \theta) = f\left(\sum_{j=1}^{q_1} h_j^{[1]} w_{ij} + b_i\right), \quad (3)$$

where

$$f(x) = \frac{1}{1+\exp\{-x\}} \quad (4)$$

is a logistic sigmoid function.

Based on this property the parameters can be learned by maximizing the log-likelihood

$$\mathscr{L}(\theta) = \sum_{n=1}^{N} \log \sum_{\mathbf{h}} p(\mathbf{v}^{(n)}, \mathbf{h} \mid \theta)$$

given N training samples $\{\mathbf{v}^{(n)}\}_{n=1,\ldots,N}$, where

$$\mathbf{h} = \left[\mathbf{h}^{[1]}; \cdots; \mathbf{h}^{[L]}\right].$$

The gradient is stochastically estimated by taking the partial derivative of the log-likelihood function $\mathscr{L}$ with respect to each parameter θ using only a subset of training samples. This estimate is then used to update the parameters, effectively forming a stochastic gradient ascent method. A standard way of computing gradient results in the following update rule for each parameter θ:

$$\nabla_\theta \mathscr{L} = \left\langle -\frac{\partial E(\mathbf{v}^{(n)}, \mathbf{h} \mid \theta)}{\partial \theta} \right\rangle_{\mathrm{d}} - \left\langle -\frac{\partial E(\mathbf{v}, \mathbf{h} \mid \theta)}{\partial \theta} \right\rangle_{\mathrm{m}}, \quad (5)$$

where $\langle\cdot\rangle_{\mathrm{d}}$ and $\langle\cdot\rangle_{\mathrm{m}}$ denote the expectation over the data distribution $P(\mathbf{h} \mid \mathbf{v}, \theta)D(\mathbf{v})$ and the model distribution $P(\mathbf{v}, \mathbf{h} \mid \theta)$, respectively [7]. D denotes an empirical data distribution.

3 Training Deep Boltzmann Machines

Although the update rule in Eq. (5) is well defined, it is intractable to compute both terms in the update rule exactly. Hence, an approach based on variational approximation together with Markov chain Monte Carlo (MCMC) sampling was proposed in [33].

First, the variational approximation is used to compute the expectation over the data distribution. It starts by approximating the posterior distribution over the hidden variables $p(\mathbf{h} \mid \mathbf{v}, \theta)$, which is intractable unless $L = 1$, by a factorial distribution

$$Q(\mathbf{h}) = \prod_{l=1}^{L} \prod_{j=1}^{N_l} \left(\mu_j^{[l]}\right)^{h_j^{[l]}} \left(1 - \mu_j^{[l]}\right)^{1-h_j^{[l]}}.$$

Each variational parameter $\mu_j^{(l)}$ can then be estimated by the following fixed-point equation:

$$\mu_j^{[l]} \leftarrow f\left(\sum_{i=1}^{N_{l-1}} \mu_i^{[l-1]} w_{ij}^{[l-1]} + \sum_{k=1}^{N_{l+1}} \mu_k^{[l+1]} w_{kj}^{[l]} + c_j^{[l]}\right), \tag{6}$$

where f is a logistic sigmoid function in Eq. (4). Note that $\mu_i^{(0)}$ is fixed to v_i and the update rule for the top layer does not contain the second summation term, that is $N_{L+1} = 0$.

This variational approximation provides the values of variational parameters that maximize the following variational lower-bound (right-hand side) of the true log-probability of $\mathbf{v}$ with respect to the current parameters θ:

$$\log p(\mathbf{v} \mid \theta) \geq \mathbb{E}_{Q(\mathbf{h})}\left[-E(\mathbf{v}, \mathbf{h})\right] + \mathscr{H}(Q) - \log Z(\theta), \tag{7}$$

where

$$\mathscr{H}(Q) = -\sum_{l=1}^{L} \sum_{j=1}^{N_l} \left(\mu_j^{[l]} \log \mu_j^{[l]} + (1 - \mu_j^{[l]}) \log(1 - \mu_j^{[l]})\right)$$

is an entropy functional.

Due to the layered structure of a DBM, it is possible to analytically sum out the odd-numbered hidden layers (see, e.g., [17]). If we denote the odd-numbered and even-numbered hidden layers by $\mathbf{h}_+$ and $\mathbf{h}_-$ respectively, we may, then, rewrite Eq. (7) into

$$\log p(\mathbf{v} \mid \theta) \geq \mathbb{E}_{Q(\mathbf{h}_-)}\left[-\sum_{\mathbf{h}_+} E(\mathbf{v}, \mathbf{h}_+, \mathbf{h}_-)\right] + \mathscr{H}(Q) - \log Z(\theta). \tag{8}$$

Since the first term of the gradient in Eq. (5) is approximated in this way, each gradient update step does not increase the true log-likelihood directly but its variational lower-bound.

Second, the expectation over the model distribution is computed by persistent sampling (see, for example, [40]). By persistent sampling, we mean that we do not wait for a sampling chain to converge before computing each update direction, but run the chain for only a few steps and continue using the same chain over consecutive updates. The simplest approach is to use Gibbs sampling, while there have been

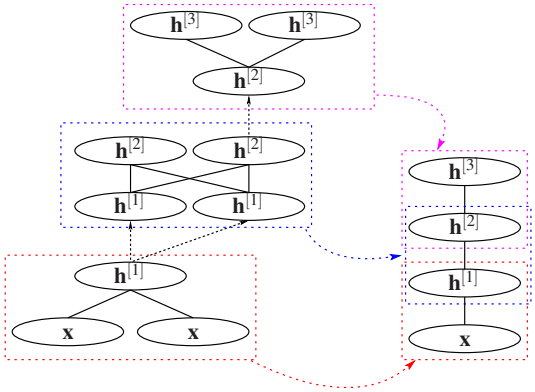

Fig. 1 Illustration of the layer-wise pretraining of a deep Boltzmann machine. The dashed directed lines indicate *copying* either the pretrained models or the activations of the hidden units of the pretrained models.

some work in applying more advanced sampling methods [32, 31, 14, 8]. In this paper, we use coupled adaptive simulated tempering (CAST) which was recently proposed in [32].

This approach closely resembles variational expectation-maximization (EM) algorithm (see, for example, [5]). Learning proceeds by alternating between finding the variational parameters μ and updating the DBM parameters θ to maximize the given variational lower-bound using the stochastic gradient method. However, it has been known and will be shown in the experiments in this paper that training a DBM using this approach starting from randomly initialized parameters is not trivial [33, 13, 11].

3.1 Layer-wise Pretraining

In [33] a pretraining algorithm to initialize the parameters of DBMs was proposed. The proposed pretraining algorithm greedily trains each layer of a DBM by considering each layer as an RBM, similarly to a pretraining approach used for training deep belief networks (DBN) [19]. However, the pretraining algorithm for DBMs differs from that for DBNs due to the undirectedness of all the edges in a DBM, which requires that the pretraining algorithm for DBMs must take into account that each hidden unit in a DBM receives a signal from both upper and lower layers (see Eq. (2)).

The algorithm proposed in [33] modifies the structure of RBMs to cope with this difference. For the bottom two layers, an RBM is modified to have two copies of visible units with tied weights such that the additional set of visible units supplies signal that compensates for the lack of signal from the second hidden layer. Similarly, an RBM that consists of the top two layers has the two copies of hidden

units. For any pair of intermediate hidden layers, an RBM is constructed to have two copies of both visible and hidden units. See Fig. 1 for an illustration.

Recently, in [35] the same authors were able to show that the variational lower bound is guaranteed to increase by adding the top hidden layer using the proposed pretraining scheme. Their proof, however, only applies to the top layer, which means that the guarantee only works for pretraining a DBM having two hidden layers.

4 Restricted Boltzmann Machines and Denoising Autoencoders

Here we briefly discuss restricted Boltzmann machines (RBM) and single-layer denoising autoencoders (DAE) which will constitute an important part of the two-stage pretraining algorithm that will be described in the next section.

An RBM is a special case of DBMs, where the number of hidden layers is restricted to one, $L = 1$ [39]. Due to this restriction it is possible to compute the posterior distribution over the hidden units conditioned on the visible neurons exactly and tractably. The conditional probability of each hidden unit $h_j = h_j^{[1]}$ is

$$p(h_j = 1 \mid \mathbf{v}, \theta) = f\left(\sum_i w_{ij} v_i + c_j\right), \tag{9}$$

where f is a logistic sigmoid function from Eq. (4).

This allows the exact and efficient computation of the positive part of the gradient in (5). However, the negative part, which is computed over the model distribution, still relies on persistent sampling, or more approximate methods such as contrastive divergence (CD) [17].

There have been extensive research on improving training RBMs using various techniques. In [9, 10] the authors proposed the enhanced gradient which exploits the fact that the gradient update of RBMs is not invariant to the bit-flipping transformation and showed that it outperforms the traditional gradient. In [14, 31, 8], advanced sampling methods for computing the negative part of the gradient based on tempering were proposed and shown to improve and stabilize learning.

A single-layer DAE is a special form of multi-layer perception network with a single hidden layer and a tied set of weights [42]. A DAE is a network that reconstructs a corrupted input vector as well as possible by minimizing the following cost function

$$\sum_{n=1}^{N} \left\| g_v\left(\mathbf{W} g_h\left(\mathbf{W}^\top \phi(\mathbf{v}^{(n)})\right)\right) - \mathbf{v}^{(n)} \right\|^2,$$

where $g_h(\cdot)$ and $g_v(\cdot)$ are component-wise nonlinear functions. ϕ is a stochastic corruption process that corrupts the input $\mathbf{v}^{(n)}$ stochastically.

There is an important difference in training DAEs compared with training RBMs. In DAEs, the objective of learning is not to learn a distribution but to minimize the

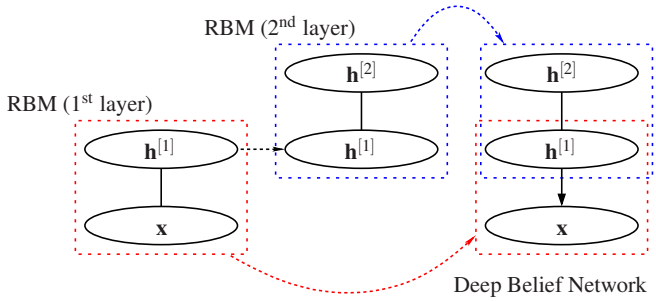

Fig. 2 Illustration of the stack of RBMs. The dashed directed lines indicate *copying* of either pretrained models or the activations of the hidden units of the pretrained models.

reconstruction error. This does not require computing a computationally intractable normalizing constant, which often leads to easier learning.[1]

These two models are important, because they can be stacked to form more powerful hierarchical models [19, 18, 2].

A deep belief network (DBN) is constructed by stacking multiple layers of RBMs [19], and a stacked DAE (sDAE) can be built by stacking DAEs on top of each other [43]. With probabilistic interpretation, one may consider these stacked models as having multiple layers of latent random variables where their posterior distributions can be computed by recursively obtaining (approximate) posterior means of the hidden units layer-wise. See Fig. 2 for the illustration.

5 A Two-Stage Pretraining Algorithm

In this paper, we propose an alternative way of initializing parameters of a DBM compared with the one described in Section 3.1. Unlike the conventional pretraining strategy we employ an approach that obtains approximate posterior distributions over hidden units and initializes parameters separately.

Before proceeding to the description of the proposed algorithm, we first divide the hidden layers of a DBM into two sets as we have done in Section 3. Let us denote a vector of hidden units in the odd-numbered layers as $\mathbf{h}_+$ and the respective vector in the even-numbered layers as $\mathbf{h}_-$. Note that due to the structure of DBMs, it is possible to explicitly sum out $\mathbf{h}_+$, which halves the space of hidden variables. Similarly we define μ_+ and μ_- as variational parameters of the hidden units in the odd-numbered layers and the even-number layers, respectively.

[1] Despite this difference in the learning objective, recent research such as [3] suggests that the DAE approximates a data generating distribution as well.

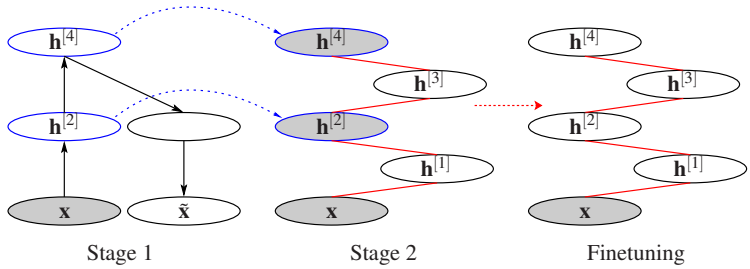

Fig. 3 Illustration of the two-stage pretraining of a deep Boltzmann machine. The dashed directed lines indicate *copying* of either pretrained models or the activations of the hidden units of the pretrained models. In this figure, a deep autoencoder is used to learn an arbitrary approximate posterior in the first stage. The red-colored edges indicate that the weights parameters learned in the second stage are used as initial values when finetuning the DBM. Note that the parameters learned in the first stage are discarded immediately after the first stage.

5.1 Stage 1

During the first stage we focus on finding a good set of variational parameters $\mu_{\_}$ of $Q(\mathbf{h}_{\_})$ that has a potential to give a reasonably high variational lower-bound in Eq. (7). In other words, we propose to first find a good posterior distribution over hidden units given a visible vector regardless of parameter values of a DBM.

Although it might sound unreasonable to find a good set of variational parameters without any fixed parameter values, we can do this by *borrowing* posterior distributions over latent variables from another model.[2]

We propose here to utilize either a DBN or a sDAE, described in Sec. 4, to find good approximate posterior distributions over hidden units in the even-numbered hidden layers. However, it is possible to use any model that finds a good binary hierarchical posterior distribution.

DBNs and sDAE's described in Sec. 4 are natural choices to find a good approximate posterior distribution over units in the even-numbered hidden layers. One justification for using either of them is that they can be trained efficiently and well (see, e.g., [1] and references therein). It is rather a trivial task where one iteratively trains each even-numbered layer as either an RBM or a DAE on top of each other.

5.2 Stage 2

Once a set of initial variational parameters $\mu_{\_}$ is found from a DBN or an sDAE, we train a model that has predictive power of the variational parameters given a visible

[2] A similar approach of borrowing the posterior means of hidden variables has been proposed in [20]. The authors of that paper initialized the variational Bayesian nonlinear blind source separation model with the posterior distribution borrowed from kernel principal component analysis.

observation. It can be simply done by letting an RBM learn a joint distribution of $\mathbf{v}$ and μ_-. In other words, we train an RBM on a set of data samples, each of which is a concatenation of $\mathbf{v}$ and μ_-.

The structure of the RBM is directly derived from the DBM such that the visible layer of the RBM corresponds to the visible layer and the even-numbered hidden layers of the DBM and the hidden layer to the odd-numbered hidden layers of the DBM. The connections between them can also follow those of the DBM. This corresponds to finding a set of DBM parameters that *fit* the variational parameters obtained in the first stage.

One way to understand what happens during the second stage is to consider what an RBM has been trained for. If we assume that we use actual samples from $Q(\mathbf{h}_-)$ instead of the variational parameters μ_-, training the RBM maximizes

$$\mathscr{L}_2(\theta) = \log \sum_{\mathbf{h}_+} \mathbb{E}_{Q(\mathbf{h}_-)} \exp\{-E(\mathbf{v}, \mathbf{h}_-, \mathbf{h}_+)\} - \log Z(\theta). \tag{10}$$

However, since we use the variational parameters which are the mean of $Q(\mathbf{h}_-)$, the actual quantity being maximized is the lower-bound of Eq. (10) according to the Jensen's inequality and the linearity of the expectation:

$$\mathscr{L}_2(\theta) \geq \sum_{\mathbf{h}_+} \left\{-E(\mathbf{v}, \mu_-, \mathbf{h}_+)\right\} - \log Z(\theta). \tag{11}$$

It is easy to see that this corresponds to maximizing the variational lower-bound of the DBM, if we group the three terms of Eq. (8) such that

$$\log p(\mathbf{v} \mid \theta) \geq \underbrace{\mathbb{E}_{Q(\mathbf{h}_-)} \left[-\sum_{\mathbf{h}_+} E(\mathbf{v}, \mathbf{h}_+, \mathbf{h}_-) \right] - \log Z(\theta)}_{\text{(a)}} + \mathscr{H}(Q).$$

The two terms grouped as (a) in the above equation is equivalent to Eq. (11) which is maximized during the second stage.[3]

Once the RBM has been trained, we can use the learned parameters as initializations for training the DBM, which corresponds to freeing $\mathbf{h}_-$ from its variational posterior distribution obtained in the first stage. Finetuning of the initialized parameters can be performed according to the standard procedure given in Sec. 3.

The overall steps of the proposed algorithm are presented in Alg. 1, and a simple illustration is given in Fig. 3.

[3] The entropy functional in Eq. (8) can be ignored, as it is constant with respect to the parameters θ.

Algorithm 1 Two-Stage Pretraining Algorithm

Input Training data $\mathbf{X}_{N\times D}$, the number of layers L and the number of units in each layer $N_1,\ldots,N_L$
$\mathbf{Q}=\mathbf{X}$
$\mathbf{X}_{-}=[]$
for $l=1\rightarrow L$ **do**
  if odd l **then**
    continue
  end if
  Train a DAE/RBM with N_l hidden units with $\mathbf{Q}$
  Set $\mathbf{Q}$ to $Q(\mathbf{h})$ from the DAE/RBM
  Append $\mathbf{Q}$ to $\mathbf{X}_{-}$
end for
Train an RBM with $\sum_{\text{even } l} N_l$ hidden units with $\mathbf{X}_{-}$
Return parameters θ of the trained RBM

5.3 Discussion

It is quite easy to see that the proposed algorithm has a high degree of freedom to plug in alternative algorithms and models in both the stages.

The most noticeable flexibility can be found in Stage 1. Any other machine learning model that gives reasonable posterior distributions over multiple layers of binary hidden units can be used instead of DBNs or sDAE's. For instance, a stack of recently proposed variants of RBMs such as spike-and-slab RBMs [12, 24] can be used.[4] Also, instead of stacking each layer at a time, one could opt to train deep autoencoders at once using recently proposed learning algorithms for feedforward neural networks (see, for instance, [28, 29, 38, 26]).

In Stage 2, one may use a DAE instead of an RBM. It will make learning faster and therefore leave more time for finetuning the model afterward. Also, the use of different algorithms for training an RBM can be considered. For quicker pretraining, one may use contrastive divergence [17] with only a small number of Gibbs sampling steps per update, or for better initial models, tempering-based advanced MCMC sampling methods such as parallel tempering [14, 8] or tempered transition [31] could be used.

Another obvious possibility is to utilize the conventional pretraining algorithm proposed in [33] during the first stage. This approach gives approximate posterior distributions over all hidden units $[\mathbf{h}_{-};\mathbf{h}_{+}]$ as well as initial values of the parameters that may be used during the second stage. In this way, one may use either an RBM or a fully visible BM (FVBM) during the second stage starting from the initialized parameters. When an RBM is used in the second stage, one could simply discard μ_{+}.

[4] One potential restriction on the choice of a single-hidden-layer neural network is that it must be computationally inexpensive and easy to compute the posterior distribution over the hidden variables.

	Stage 1	Stage 2	Finetuning
DBM	×	×	DBM
$\text{DBM}^{\text{sDAE}}_{\text{RBM}}$	sDAE	RBM	DBM
$\text{DBM}^{\text{DBN}}_{\text{RBM}}$	DBN	RBM	DBM
$\text{DBM}^{\text{S\&H}}$	(S)	×	DBM
$\text{DBM}^{\text{S\&H}}_{\text{RBM}}$	(S)	RBM	DBM
$\text{DBM}^{\text{S\&H}}_{\text{FVBM}}$	(S)	FVBM	DBM

Table 1 Algorithms used in the experiment. (S) – the pretraining algorithm from [33].

One important point of the proposed algorithm is that it provides another research perspective in training DBMs. The existing pretraining scheme developed in [33, 34] was based on the observation that under certain assumptions the variational lower-bound could be increased by learning weight parameters layer wise. However, the success of the proposed scheme suggests that it may not be the set of parameters that need to be pretrained, but the set of variational parameters that determine how tight the variational lower-bound is and their corresponding parameters. This way of approaching the problem of training DBMs enables us to potentially find another explanation on why training large DBMs without pretraining is not trivial.

6 Experiments

In the experiments, we train DBMs on two datasets which are a handwritten digit dataset (MNIST) [23] and Caltech-101 Silhouettes dataset [25]. We used the MNIST and Caltech-101 Silhouettes datasets because experimental results of using DBMs for both datasets are readily available for direct comparison [36, 32, 27].

We train DBMs with varying numbers of units in the hidden layers; 500-1000, 500-500-1000, 500-500-500-1000. The first two architectures were used in [36, 32], which enables us to directly compare our proposed algorithm with the conventional pretraining algorithm.

For learning algorithms, we extensively tried various combinations. They are presented in Table 1. In summary, a $DBM^{stage\ 1}_{stage\ 2}$ denotes a deep Boltzmann machine in which its superscript and subscript denote the algorithms used during the first and second stages, respectively.

We used contrastive divergence (CD) to train RBMs in the first stage, and the persistent CD [41] with coupled adaptive simulated annealing (CAST) was used in the second stage. DAEs were trained using stochastic gradient descent (SGD) algorithm with backpropagation [30].

When a DBM was finetuned, we estimated the variational parameters by the variational approximation with at most 30 mean-field updates (see Eq. (6)). The model statistics, the negative part of the gradient, was computed by CAST. When the conventional pretraining algorithm is used, we do not explicitly make duplicate copies of visible or hidden units, but only double the corresponding weight parameters [34].

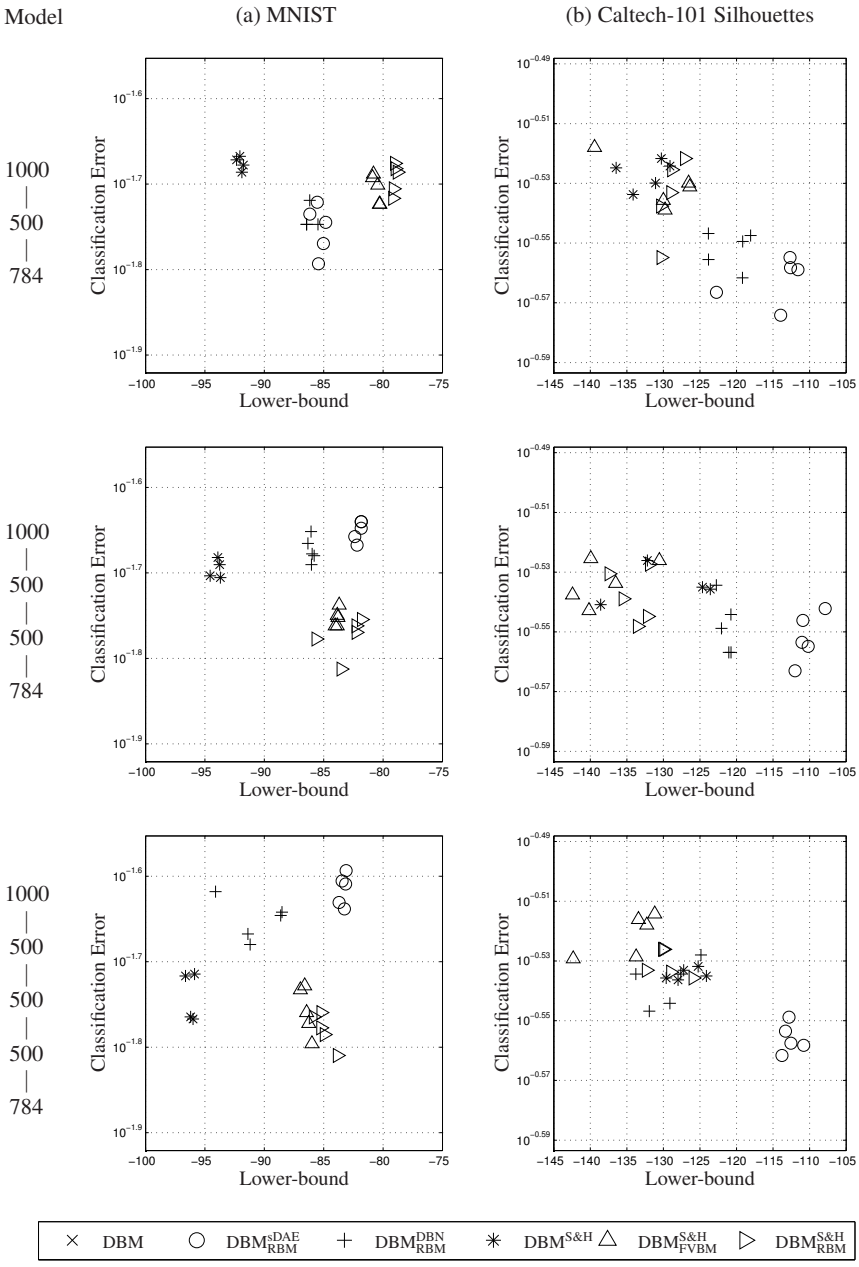

Fig. 4 Performance of the trained DBMs. Best performing models are in the bottom right corners.

We trained each model for 200 epochs in the case of MNIST and 2000 epochs in the case of Caltech-101 Silhouettes with a learning rate scheduled by $\frac{\eta_0}{1+\frac{n}{5000}}$ where n is the number of updates. η_0 was set to 0.01 and 0.0005 for pretraining and finetuning, respectively. When we did not pretrain a DBM, we trained each DBM for twice more epochs and set η_0 to 0.001. In all cases, we used a minibatch of size 128.

When training RBMs with CAST, we used equally spaced 21 tempered fast chains from 0.9 to 1 with a single sampling step each update. When DAEs were trained, at every update we dropped off a randomly chosen set of hidden units with probability 0.1 [16]. During the finetuning stage, we used CAST with the equally spaced 21 tempered chains from 0.9 to 1.

We evaluated the resulting models with the variational lower-bound of log-probabilities and the classification errors of test samples. The variational lower-bounds reflect the generative performance of the model. We trained a linear SVM for each hidden layer l using μ_l as its features to compute the classification errors. This is expected to show how much discriminative information about input samples is captured by each hidden layer of the model.

The intractable normalization constant ($\log Z(\theta)$ in Eq. (7)) required when computing the variational lower-bound was approximated using annealed importance sampling (AIS) [37]. For each model, we used 20001 equally spaced tempered chains from 0 to 1 with 128 independent runs.

All models were trained five times starting from different random initializations. We report the medians over these random trials.

6.1 Result and Analysis

Fig. 4 presents the result using both the lower-bound of log-probabilities and the classification error of the test samples. As has already been expected, none of the models trained *without* pretraining have been able to perform well enough to be presented inside the boundaries of the boxes in Fig. 4.

It is clear from the figures that the proposed two-stage pretraining algorithm outperforms, in all cases, the conventional pretraining algorithm ($\mathrm{DBM}^{\mathrm{S\&H}}$). On MNIST, the DBMs pretrained with the proposed algorithm using the conventional pretraining algorithm in the first stage achieved the best performance. In the case of Caltech-101 Silhouettes, $\mathrm{DBM}^{\mathrm{sDAE}}_{\mathrm{RBM}}$ was able to achieve superior performance in both generative and discriminative modeling. It is notable that without any pretraining (DBM) we were not able to achieve any reasonable performance.

Fig. 5 presents layer-wise classification errors. It is clear from the significantly lower accuracies in the higher hidden layers of the DBMs trained without pretraining that pretraining is essential to allow upper layers to capture discriminative struc-

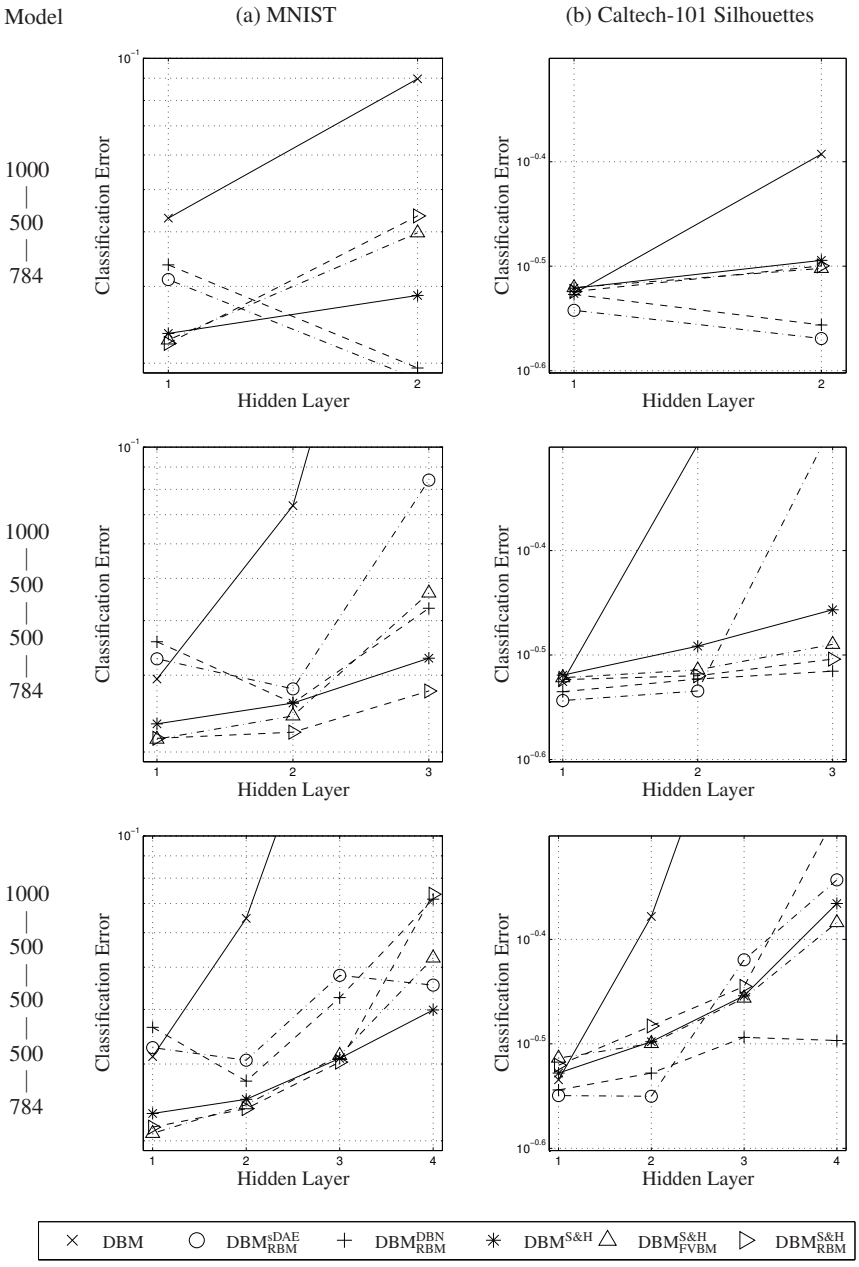

Fig. 5 Layer-wise Discriminative Performance. Lower is better.

tures of data. $\text{DBM}^{\text{DBN}}_{\text{RBM}}$ and $\text{DBM}^{\text{S\&H}}_{\text{RBM}}$ were most effective in ensuring the upper hidden layers to have better discriminative property.[5]

7 Conclusions

The experimental success of the proposed two-stage pretraining algorithm in training DBMs suggests that the difficulty of DBM learning might be due to the fact that the estimated variational lower-bound at the initial stage of learning is too crude, or too loose. Once the variational parameters are initialized well with another deep hierarchical model, the parameters of a DBM may be fitted to give a tighter variational lower-bound which facilitates jointly estimating all parameters.

The proposed two-stage pretraining algorithm provides a general framework in which many hierarchical deep learning models can be used. It even makes possible to include the conventional pretraining algorithm as a part of the proposed algorithm and improve upon it. This is a significant step in developing and improving a training algorithm for DBMs, as it allows us to fully utilize other learning algorithms that have been extensively studied previously.

7.1 Future Work

Recently, two additional algorithms for training DBMs have been proposed. In [27] the authors proposed to center the activations of the neurons in a DBM, which is closely related to the previously proposed method of the enhanced gradient for RBMs [9, 10]. They showed that this simply method allows training a DBM without any pretraining, though, without any direct comparison to the method of pretraining.

The authors of [15] and [6] proposed an alternative learning criterion based on the idea of generalized pseudo-likelihood [21]. The alternative criterion does *not* maximize the log-likelihood but maximizes the predictive conditional probabilities among all the visible variables of a DBM.

It is important in the future to compare these recently proposed algorithms together with the two-stage pretraining algorithm as well as the conventional pretraining algorithm against each other. In the case of RBMs, in [25], the authors compared various learning criteria such as maximum-likelihood, contrastive divergence, ratio matching [22] and maximum pseudo-likelihood [4], and they found that each algorithm resulted in solutions that are different in multiple aspects. A similar approach of comparing different learning criteria for the DBM will reveal their inductive biases and allow us to choose an appropriate learning algorithm.

[5] It should be noted that these accuracies were computed purely to illustrate the effect of generative training of DBMs , and the reported accuracies are lower than other state-of-the-art accuracies (see, [15] for state-of-the-art accuracies for MNIST and [9] for Caltech-101 Silhouettes).

Acknowledgements This work was supported by "the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170)".

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(8), 1798–1828 (2013)
2. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19, pp. 153–160. MIT Press, Cambridge, MA (2007)
3. Bengio, Y., Yao, L., Alain, G., Vincent, P.: Generalized denoising auto-encoders as generative models. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 26, pp. 899–907 (2013)
4. Besag, J.: Statistical Analysis of Non-Lattice Data. Journal of the Royal Statistical Society. Series D (The Statistician) 24(3), 179–195 (1975)
5. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
6. Brakel, P., Stroobandt, D., Schrauwen, B.: Training energy-based models for time-series imputation. Journal of Machine Learning Research 14, 2771–2797 (2013)
7. Cho, K.: Improved Learning Algorithms for Restricted Boltzmann Machines. Master's thesis, Aalto University School of Science (2011)
8. Cho, K., Raiko, T., Ilin, A.: Parallel tempering is efficient for learning restricted boltzmann machines. In: Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN 2010). pp. 1–8 (Jul 2010)
9. Cho, K., Raiko, T., Ilin, A.: Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011). pp. 105–112. ACM, New York, NY, USA (Jun 2011)
10. Cho, K., Raiko, T., Ilin, A.: Enhanced gradient for training restricted Boltzmann machines. Neural Computation 25(3), 805–831 (Mar 2013)
11. Cho, K., Raiko, T., Ilin, A.: Gaussian-bernoulli deep boltzmann machines. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2013). Texas, USA (August 2013)
12. Courville, A., Bergstra, J., Bengio, Y.: A spike and slab restricted Boltzmann machine. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011) (2011)
13. Desjardins, G., Courville, A., Bengio, Y.: On training deep Boltzmann machines. arXiv:1203.4416 [cs.NE] (Mar 2012)
14. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Delalleau, O.: Parallel tempering for training of restricted Boltzmann machines. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010). JMLR Workshop and Conference Proceedings, vol. 9, pp. 145–152. JMLR W&CP (2010)
15. Goodfellow, I., Miraz, M., Courville, A., Bengio, Y.: Multi-prediction deep Boltzmann machines. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 26. pp. 548–556 (Dec 2013)
16. Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 [cs.NE] (Jul 2012)
17. Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771–1800 (Aug 2002)
18. Hinton, G., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (Jul 2006)

19. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (Jul 2006)
20. Honkela, A., Harmeling, S., Lundqvist, L., Valpola, H.: Using kernel PCA for initialisation of variational Bayesian nonlinear blind source separation method. In: Puntonet, C., Prieto, A. (eds.) Proceedings of Independent Component Analysis and Blind Signal Separation (ICA 2004). Lecture Notes in Computer Science, vol. 3195, pp. 790–797. Springer (2004)
21. Huang, F., Ogata, Y.: Generalized pseudo-likelihood estimates for markov random fields on lattice. Annals of the Institute of Statistical Mathematics 54(1), 1–18 (2002)
22. Hyvärinen, A.: Some extensions of score matching. Computational Statistics & Data Analysis 51(5), 2499–2512 (Feb 2007)
23. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. vol. 86, pp. 2278–2324 (1998)
24. Luo, H., Carrier, P., Courville, A., Bengio, Y.: Texture modeling with convolutional spike-and-slab RBMs and deep extensions. In: Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2013). JMLR Workshop and Conference Proceedings, vol. 31, pp. 415–423. JMLR W&CP (Apr 2013)
25. Marlin, B.M., Swersky, K., Chen, B., de Freitas, N.: Inductive principles for restricted Boltzmann machine learning. In: Proceedings of the Thirteenth Internation Conference on Artificial Intelligence and Statistics (AISTATS 2010). JMLR Workshop and Conference Proceedings, vol. 9, pp. 509–516. JMLR W&CP (2010)
26. Martens, J.: Deep learning via Hessian-free optimization. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th Internation Conference on Machine Learning (ICML 2010). pp. 735–742. Haifa, Israel (Jun 2010)
27. Montavon, G., Müller, K.R.: Deep Boltzmann machines and the centering trick. In: Montavon, G., Orr, G., Mller, K.R. (eds.) Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol. 7700, pp. 621–637. Springer Berlin Heidelberg (2012)
28. Pascanu, R., Bengio, Y.: Revisiting natural gradient for deep networks. arXiv:1003.0358 [cs.NE] (2013)
29. Raiko, T., Valpola, H., LeCun, Y.: Deep learning made easier by linear transformations in perceptrons. In: Proceedings of the Fifteenth Internation Conference on Artificial Intelligence and Statistics (AISTATS 2012). JMLR Workshop and Conference Proceedings, vol. 22, pp. 924–932. JMLR W&CP (Apr 2012)
30. Rumelhart, D.E., Hinton, G., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(Oct), 533–536 (1986)
31. Salakhutdinov, R.: Learning in Markov random fields using tempered transitions. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) Advances in Neural Information Processing Systems 22, pp. 1598–1606 (2009)
32. Salakhutdinov, R.: Learning deep Boltzmann machines using adaptive MCMC. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML 2010). pp. 943–950. Haifa, Israel (Jun 2010)
33. Salakhutdinov, R., Hinton, G.: Deep Boltzmann machines. In: Proceedings of the Twelfth Internation Conference on Artificial Intelligence and Statistics (AISTATS 2009). JMLR Workshop and Conference Proceedings, vol. 5, pp. 448–455. JMLR W&CP (2009)
34. Salakhutdinov, R., Hinton, G.: A better way to pretrain deep Boltzmann machines. In: Bartlett, P., Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 25, pp. 2456–2464 (2012)
35. Salakhutdinov, R., Hinton, G.: An effcient learning procedure for deep Boltzmann machines. Neural Computation 24, 1967–2006 (2012)
36. Salakhutdinov, R., Larochelle, H.: Efficient learning of deep boltzmann machines. In: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (2011)
37. Salakhutdinov, R., Murray, I.: On the quantatitive analysis of deep belief networks. In: Proceedings of the 25th International Conference on Machine learning (ICML 2008). pp. 872–879. ACM, New York, NY, USA (2008)
38. Schulz, H., Behnke, S.: Learning two-layer contractive encodings. In: Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN 2012) (Sep 2012)

39. Smolensky, P.: Information processing in dynamical systems: foundations of harmony theory. In: Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations, pp. 194–281. MIT Press, Cambridge, MA, USA (1986)
40. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th Internation Conference on Machine Learning (ICML 2008). pp. 1064–1071. ACM, New York, NY, USA (2008)
41. Tieleman, T., Hinton, G.: Using fast weights to improve persistent contrastive divergence. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009). pp. 1033–1040. ACM, New York, NY, USA (2009)
42. Vincent, P.: A connection between score matching and denoising autoencoders. Neural Computation 23(7), 1661–1674 (Jul 2011)
43. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11, 3371–3408 (Dec 2010)