A Two-Stage Pretraining Algorithm for Deep Boltzmann Machines

KyungHyun Cho, Tapani Raiko, Alexander Ilin and Juha Karhunen

Department of Information and Computer Science
Aalto University School of Science, Finland
{firstname.lastname@aalto.fi}

Abstract. A deep Boltzmann machine (DBM) is a recently introduced Markov random field model that has multiple layers of hidden units. It has been shown empirically that it is difficult to train a DBM with approximate maximum-likelihood learning using the stochastic gradient unlike its simpler special case, restricted Boltzmann machine (RBM). In this paper, we propose a novel pretraining algorithm that consists of two stages; obtaining approximate posterior distributions over hidden units from a simpler model and maximizing the variational lower-bound given the fixed hidden posterior distributions. We show empirically that the proposed method overcomes the difficulty in training DBMs from randomly initialized parameters and results in a better, or comparable, generative model when compared to the conventional pretraining algorithm.

Keywords: Deep Boltzmann Machine, Deep Learning, Pretraining

1 Introduction

Deep Boltzmann machine (DBM), proposed in [14], is a recently introduced variant of Boltzmann machines which extends widely used restricted Boltzmann machines (RBM) to a model that has multiple hidden layers. It differs from the popular deep belief network (DBN) [5] in that every edge in the DBM model is undirected. In this way, DBMs facilitate 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, [14, 4]) 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 [14] a greedy layer-wise pretraining algorithm was proposed to be used to initialize parameters of DBMs, and it was shown that it largely overcomes the difficulty of learning a good generative model.

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

Based on this observation, our proposed method 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 fixed approximate posterior distribution, we train an RBM that learns a distribution over a combination of data samples and their corresponding posterior distributions of hidden units. Finetuning the model is then trivial as one only needs to free hidden variables from the approximate posterior distribution computed during the first stage.

We show that the proposed algorithm helps learning a good generative model which is empirically comparable to the pretraining method proposed in [14]. Furthermore, we discuss the potential degrees of freedom in extending the proposed approach.

2 Deep Boltzmann Machines

We start by describing deep Boltzmann machines (DBM) [14]. A DBM with L layers of hidden neurons is defined by the following energy function:

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

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. $\mathbf{W} = [w_{i,j}]$ is the set of weights between the visible neurons and the first layer hidden neurons, while $\mathbf{U}^{(l)} = \left[u_{j,k}^{(l)}\right]$ is the set of weights between the l-th and $l+1$-th hidden neurons. 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 $\boldsymbol{\theta}$ 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 $p(\mathbf{x}\mid\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp\{-E(\mathbf{x}\mid\boldsymbol{\theta})\}$. Based on this property the parameters can be learned by maximizing the log-likelihood $\mathcal{L} = \sum_{n=1}^{N}\log\sum_{\mathbf{h}} p(\mathbf{v}^{(n)},\mathbf{h}\mid\boldsymbol{\theta})$ 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 computed by taking the partial derivative of the log-likelihood function with respect to each parameter is used in most cases with a mini-batch per update. It 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:

$$\nabla\theta = \left\langle -\frac{\partial E(\mathbf{v}^{(n)},\mathbf{h}\mid\boldsymbol{\theta})}{\partial\theta}\right\rangle_{\mathrm{d}} - \left\langle -\frac{\partial E(\mathbf{v},\mathbf{h}\mid\boldsymbol{\theta})}{\partial\theta}\right\rangle_{\mathrm{m}}, \tag{2}$$

where $\langle\cdot\rangle_{\mathrm{d}}$ and $\langle\cdot\rangle_{\mathrm{m}}$ denote the expectation over the data distribution $P(\mathbf{h}\mid\{\mathbf{v}^{(n)}\},\boldsymbol{\theta})$ and the model distribution $P(\mathbf{v},\mathbf{h}\mid\boldsymbol{\theta})$, respectively [3].

3 Training Deep Boltzmann Machines

Although the update rules in Eq. (2) are well defined, it is intractable to compute them exactly. Hence, an approach that uses variational approximation together with Markov chain Monte Carlo (MCMC) sampling was proposed in [14].

First, the variational approximation is used to compute the expectation over the data distribution. It starts by approximating $p(\mathbf{h} \mid \mathbf{v}, \boldsymbol{\theta})$, which is intractable unless $L = 1$, by a factorial distribution $Q(\mathbf{h}) = \prod_{l=1}^{L} \prod_{j=1}^{N_l} \mu_j^{(l)}$. The variational parameters $\mu_j^{(l)}$ can then be estimated by the following fixed-point equation:

$$\mu_j^{(l)} \leftarrow f\left(\sum_{i=1}^{N_{l-1}} \mu_i^{(l-1)} w_{ij}^{(l-1)} + \sum_{k=1}^{N_{l+1}} \mu_k^{(l+1)} w_{kj}^{(l)} + c_j^{(l)}\right), \tag{3}$$

where $f(x) = \frac{1}{1+\exp\{-x\}}$. Note that $\mu_i^{(0)} = v_i$ 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 lower-bound with respect to the current parameters:

$$p(\mathbf{v} \mid \boldsymbol{\theta}) \geq \mathbb{E}_{Q(\mathbf{h})}\left[-E(\mathbf{v}, \mathbf{h})\right] + \mathcal{H}(Q) - \log Z(\boldsymbol{\theta}), \tag{4}$$

where

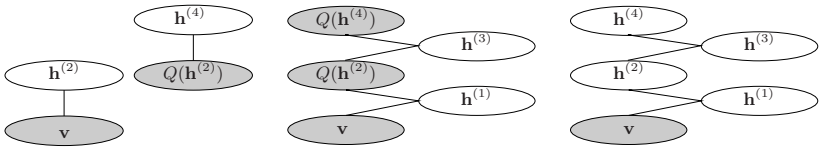
$$\mathcal{H}(Q) = -\sum_{l=1}^{L} \sum_{j=1}^{N_l} \left(\mu_j^{(l)} \log \mu_j^{(l)} + (1 - \mu_j^{(l)}) \log(1 - \mu_j^{(l)})\right)$$

is an entropy functional. Hence, each gradient update step does not increase the exact log-likelihood but its variational lower-bound.

Second, the expectation over the model distribution is computed by persistent sampling. The simplest approach is to use Gibbs sampling.

This approach closely resembles variational expectation-maximization (EM) algorithm (see, for example, [2]). Learning proceeds by alternating between finding the variational parameters $\boldsymbol{\mu}$ and updating the DBM parameters 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 [14, 4].

Hence, in [14] a pretraining algorithm to initialize the parameters of DBMs was proposed. The pretraining algorithm greedily trains each layer of a DBM by considering each layer as an RBM, following a pretraining approach used for training deep belief networks (DBN) [5]. However, due to the undirectedness of edges in DBMs it has been proposed to use the first layer RBM with two duplicate copies of visible units with tied weights and the last layer RBM with two duplicate copies of hidden units with tied weights. Once one layer has been trained, another layer can be trained on the aggregate posterior distribution of the hidden units of the lower layer to extend the depth. After the pretraining, learned weights are used as initializations of weights of DBMs.



(a) Stage 1 (b) Stage 2 (c) Finetuning

Fig. 1. Illustration of the two-stage pretraining algorithm followed by finetuning of all parameters. Shaded nodes indicate clamped variables whereas white nodes are free variables.

4 A Two-stage Pretraining Algorithm

In this paper, we propose an alternative way of initializing parameters of a DBM compared with the one described at the end of Section 3. We employ an approach that separately obtains posterior distributions over hidden units and initializes parameters.

Before proceeding to the description of the proposed algorithm, we first divide the hidden layers of a DBM into two sets. 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 $\mathbf{h}_-$. In this sense we may define $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ as variational parameters of the hidden units in the odd-numbered layers and the even-number layers, respectively.

Stage 1: We focus on finding a good set of variational parameters $\boldsymbol{\mu}_-$ of $Q(\mathbf{h}_-)$ that has a potential to give a reasonably high variational lower-bound in Eq. (4). 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 other models.

DBNs and sDAE's, described in [5] and [16], 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 becomes a trivial task as one can iteratively train each even-numbered layer as either an RBM or a DAE on top of each other, as is a common practice when a DBN or a sDAE is trained.

Stage 2: Once a set of initial variational parameters $\boldsymbol{\mu}_-$ is found from a DBN or an sDAE, we train a model that has predictive power of the variational parameters given a visible vector. It can be simply done by letting an RBM learn a joint distribution of $\mathbf{v}$ and $\boldsymbol{\mu}_-$.

The structure of the RBM can be directly derived from the DBM such that its visible layer corresponds to the visible layer and the even-numbered hidden layers of the DBM and its 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.

Once an 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.

A simple illustration of the two-stage pretraining algorithm is given in Fig. 1.

4.1 Discussion

It is quite easy to see that the proposed algorithm has high degree of freedom to plug in alternative algorithms and models in both 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 RBMs or DAEs. Also, instead of stacking each layer at a time, one could opt to train deep autoencoders at once using advanced backpropagation algorithms (see, for instance, [10]).

In Stage 2, one may opt to 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 [6] , or for better initial models, advanced MCMC sampling methods could be used.

Another obvious possibility is to utilize the conventional pretraining algorithm proposed in [14] during the first stage. This approach gives approximate posterior distributions over all hidden units as well as initial values of the parameters. 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 $\boldsymbol{\mu}_+$.

One important point of the proposed algorithm is that it provides another research perspective in training DBMs. The existing pretraining scheme developed in [14, 11] 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 directly pretrained, but the set of variational parameters that determine how tight the variational lower-bound is and their corresponding parameters.

5 Experiments

In the experiments, we train DBMs on two datasets which are a handwritten digit dataset (MNIST) [7] and Caltech-101 Silhouettes dataset [8]. We used the MNIST and Caltech-101 Silhouettes datasets because experimental results of using DBMs for both datasets are readily available for direct comparison [13, 12, 9].

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 [13, 12], 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^{stage1}_{stage2} 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 [15] with coupled adaptive simulated annealing (CAST) was used in the second stage. DAEs were trained using stochastic backpropagation algorithm.

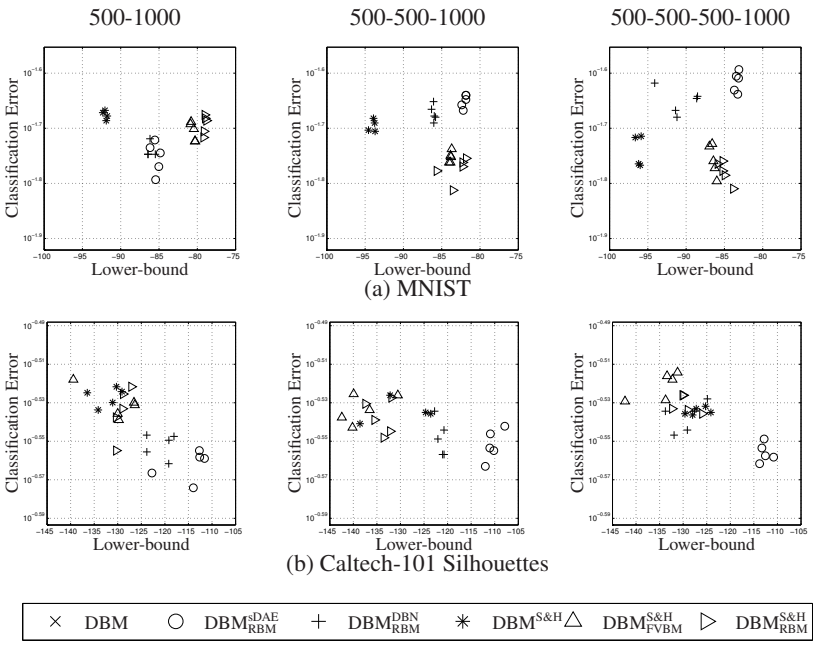


Fig. 2. Performance of the trained DBMs. Best performing models are in bottom right corners of each plot.

When a DBM was finetuned, we estimated the variational parameters by running at most 30 mean-field fixed-point updates. The model statistics, the negative part of the gradient, was computed by CAST.

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

Table 1. Algorithms used in the experiment. (S) – the pretraining algorithm from [14].

We evaluated the resulting models with the variational lower-bound of log-probabilities and the classification error of test samples. The variational lower-bounds reflect the generative performance of the model. The classification accuracy computed from a linear support vector machine (SVM) tells us the discriminative property of the hidden units. We trained a linear SVM for each hidden layer l using $\boldsymbol{\mu}_l$ as its features. This is expected to show how much information about input samples is captured by each hidden layer of the model.

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

5.1 Result and Analysis

Fig. 2 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

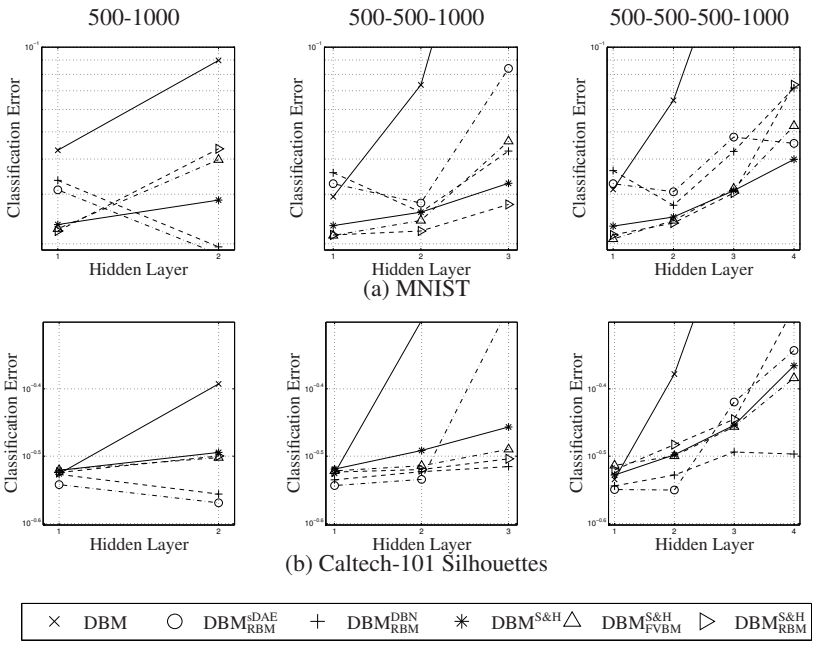


Fig. 3. Layer-wise Discriminative Performance. Lower is better.

trained *without* pretraining have been able to perform well enough to be presented inside the boundaries of the boxes in Fig. 2.

It is clear from the figures that the proposed two-stage pretraining algorithm outperforms, in all cases, the conventional pretraining algorithm (DBM$^{S\&H}$). 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, DBM$^{sDAE}_{RBM}$ 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. 3 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 structures of data. DBM$^{DBN}_{RBM}$ and DBM$^{S\&H}_{RBM}$ were most effective in ensuring the upper hidden layers to have better discriminative property.

6 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 one initializes the variational parameters well enough by utilizing another deep

hierarchical model, the parameters of a DBM can 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.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation Learning: A Review and New Perspectives. arXiv:1206.5538 [cs.LG] (Jun 2012)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, corrected 2nd printing edn. (2007)
3. Cho, K.: Improved Learning Algorithms for Restricted Boltzmann Machines. Master's thesis, Aalto University School of Science (2011)
4. Desjardins, G., Courville, A., Bengio, Y.: On training deep Boltzmann machines. arXiv:1203.4416 [cs.NE] (Mar 2012)
5. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (July 2006)
6. Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771–1800 (August 2002)
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp. 2278–2324. No. 11
8. Marlin, B.M., Swersky, K., Chen, B., de Freitas, N.: Inductive principles for restricted Boltzmann machine learning. In: Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2010). pp. 509–516 (2010)
9. Montavon, G., Müller, K.R.: Deep Boltzmann machines and the centering trick. In: Montavon, G., Orr, G.B., Müller, K.R. (eds.) Neural Networks: Tricks of the trade, Reloaded, LNCS, vol. 7700. Springer, 2nd edn. (2012)
10. Raiko, T., Valpola, H., LeCun, Y.: Deep learning made easier by linear transformations in perceptrons. In: Proc. of the 15th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2012). La Palma, Canary Islands, Spain (April 2012)
11. Salakhutdinov, R., Hinton, G.E.: A Better Way to Pre-Train Deep Boltzmann Machines. In: Advances in Neural Information Processing Systems (2012)
12. Salakhutdinov, R.: Learning deep Boltzmann machines using adaptive MCMC. In: Fürnkranz, J., Joachims, T. (eds.) Proc. of the 27th Int. Conf. on Machine Learning (ICML 2010). pp. 943–950. Omnipress, Haifa, Israel (June 2010)
13. Salakhutdinov, R., Hinton, G.: An efficient learning procedure for deep Boltzmann machines. Tech. Rep. MIT-CSAIL-TR-2010-037, MIT (August 2010)
14. Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: Proc. of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2009). pp. 448–455 (2009)
15. Tieleman, T., Hinton, G.E.: Using fast weights to improve persistent contrastive divergence. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 1033–1040. ICML '09, ACM, New York, NY, USA (2009)
16. 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)