Joint Training of Partially-Directed Deep Boltzmann Machines

Ian J. Goodfellow
goodfeli@iro.umontreal.ca

Aaron Courville
aaron.courville@umontreal.ca

Yoshua Bengio

Département d'Informatique et de Recherche Opérationelle
Université de Montréal
Montréal, QC

Abstract

We introduce a deep probabilistic model which we call the *partially directed deep Boltzmann machine* (PD-DBM). The PD-DBM is a model of real-valued data based on the deep Boltzmann machine (DBM) and the spike-and-slab sparse coding (S3C) model. We offer a hypothesis for why DBMs may not be trained succesfully without greedy layerwise training, and motivate the PD-DBM as a modified DBM that can be trained jointly.

1 Introduction

We propose a new deep probabilistic model of real-valued data called the *partially directed deep Boltzmann machine* (PD-DBM), based on the deep Boltzmann machine (DBM) (Salakhutdinov and Hinton, 2009) and spike-and-slab sparse coding (S3C) (Goodfellow *et al.*, 2012).

Most previous deep probabilistic models require an initial greedy, layer-by-layer training stage. We consider the particular case of the deep Boltzmann machine and offer a hypothesis for why it must be trained greedily. Based on this hypothesis, we motivate the PD-DBM as a modified version of the DBM that admits joint training of a two-layer version.

We demonstrate that our deep model can succesfully be jointly trained to generate samples from the correct distribution, and that it improves significantly over the single layer version. We also analyize the parameters of the model to verify that the second layer is being used as we intend.

These results motivate further exploration of the PD-DBM model, and suggest further work to investigate whether our hypothesis can lead to further successful modifications of the DBM training algorithm.

2 Related work

Our model draws inspiration primarily from two pre-existing models: the deep Boltzmann machine and spike-and-slab sparse coding. We describe these models, then review past attempts at jointly training deep models.

2.1 Deep Boltzmann machines

The deep Boltzmann machine (Salakhutdinov and Hinton, 2009) is defined broadly but in practice researchers use a variant of it that is organized into layers, with units inside of a layer con-

ditionally independent from each other given the neighboring layers. This most commonly used variant of the DBM consists of an observed input vector $v \in \{0,1\}^D$, and a set of binary vectors $\boldsymbol{h} = \{h^{(1)}, \ldots, h^{(L)}\}$ where $h^{(l)} \in \{0,1\}^{N_l}$ and L is the number of hidden layers. The DBM defines the following probability distribution:

$$P_{\text{DBM}}(v, \boldsymbol{h}) \propto \exp\left(-b^{(0)T}v - v^T W^{(1)} h - \sum_{l=1}^{L} b^{(l)T} h^{(l)} - \sum_{l=2}^{L} h^{(l-1)T} W^{(l)} h^{(l)}\right).$$

where each W is a matrix of weights and each b is a vector of biases.

Training a deep Boltzmann machine requires both a variational approximation due to the intractable model posterior and a sampling approximation to the gradient of the log partition function.

Deep Boltzmann machines have primarily been used for modeling images and extracting features for object recognition (Salakhutdinov and Hinton, 2009; Salakhutdinov and Larochelle, 2010; Salakhutdinov *et al.*, 2010) . The deep Boltzmann machine is successful as both a generative model and a feature extractor.

2.2 Spike-and-slab sparse coding

The spike-and-slab sparse coding (S3C) model is a well-studied model of natural images. It consists of latent binary *spike* variables $h \in \{0,1\}^N$, latent real-valued *slab* variables $s \in \mathbb{R}^N$, and real-valued visible vector $v \in \mathbb{R}^D$ generated according to this process:

$\forall i \in \{1, \ldots, N\}, d \in \{1, \ldots, D\}$,

$$\begin{aligned} p(h_i = 1) &= \sigma(b_i) \\ p(s_i \mid h_i) &= \mathcal{N}(s_i \mid h_i \mu_i, \alpha_{ii}^{-1}) \\ p(v_d \mid s, h) &= \mathcal{N}(v_d \mid W_{d:}(h \circ s), \beta_{dd}^{-1}) \end{aligned} \tag{1}$$

where σ is the logistic sigmoid function, b is a set of biases on the spike variables, μ and W govern the linear dependence of s on h and v on s respectively, α and β are diagonal precision matrices of their respective conditionals, and $h \circ s$ denotes the element-wise product of h and s.

We refer to the variables h_i and s_i as jointly defining the i^{th} hidden unit, so that there are a total of N rather than $2N$ hidden units. The state of a hidden unit is best understood as $h_i s_i$, that is, the spike variables gate the slab variables[1].

The model has previously been used as a model of V1 cortex (Garrigues and Olshausen, 2008), for source separation (Lücke and Sheikh, 2011), for prediction of missing features (Zhou *et al.*, 2009; Mohamed *et al.*, 2012), denoising and compressed sensing (Zhou *et al.*, 2009), as a component of Gaussian process regression models (Titsias and Lázaro-Gredilla, 2011), and as a feature extractor for object recognition (Goodfellow *et al.*, 2012).

The S3C model has an intractable posterior but a tractable partition function. It can thus be fit by maximizing a variational bound on the log likelihood, without need to approximate the gradient of the log partition function.

While the S3C model has been established as a good feature extractor and even as a good enough density model to be used for inpainting and denoising, its factorial prior makes it a poor generative model.

2.3 Greedy pretraining versus joint training

Deep models are commonly pretrained in a greedy layerwise fashion. For example, a DBM is usually initialized by modifying a stack of RBMs, with one RBM trained on the data and each of the other RBMs trained on samples of the previous RBM's hidden layer.

[1] We can essentially recover h_i and s_i from $h_i s_i$ since $s_i = 0$ has zero measure.

Any greedy training procedure can obviously get stuck in a local minimum. Avoiding the need for greedy training could thus result in better models. For example, when pretraining with an RBM, the lack of explaining away in the posterior prevents the first layer from learning nearly parallel weight vectors, since these would result in similar activations (up to the bias term, which could simply make one unit always less active than the other). Even though the deeper layers of the DBM could implement the explaining away needed for these weight vectors to function correctly (i.e., to have the one that resembles the input the most activate, and inhibit the other unit), the greedy learning procedure does not have the opportunity to learn such weight vectors.

Previous efforts at jointly training even two layer DBMs on MNIST have failed (Salakhutdinov and Hinton, 2009; Desjardins *et al.*, 2012) Typically, the jointly trained DBM does not make good use of the second layer, either because the second layer weights are very small or because they contain several duplicate weights focused on a small subset of first layer units that became active early during training. (Montavon and Müller, 2012) has proposed a modified model that may be jointly trained, by centering the state of the units. Here we explore an alternate modified model where some of the units remain $\{0, 1\}$-valued.

3 Partially directed deep Boltzmann machines

We now introduce a model that resolves both the poor generative abilities of S3C and, in the two layer case, the difficulties that the DBM faces with joint training.

We hypothesize that DBM joint training fails because the second layer hidden units in a DBM must both learn to model correlations in the first layer induced by the data and to counteract correlations in the first layer induced by the model family.

Early in training, the second layer weights are near 0. Assuming that the second layer weights remain near 0 for a brief period while the first layer weights start to grow, the model should behave similarly to an RBM defined by the first layer. We can thus gain insight into the second layer's modeling task by considering how the first layer RBM behaves.

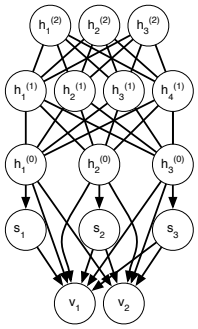

Figure 1: A graphical model depicting an example PD-DBM.

The RBM prior acts to correlate first layer hidden units that have similar weight vectors. This is because the posterior, rather than the prior, distribution over the hidden units is factorial. When the prior over the hidden units is factorial, two units with similar weight vectors will compete to explain the data, so they have will have very dissimilar activation patterns. When the posterior over the hidden units is factorial, two units with similar weight vectors will respond similarly to similar inputs. This is easiest to see in a Gaussian-Bernoulli RBM (Welling *et al.*, 2005). The energy function

$$E(v, h) = \frac{1}{2} v^T v - v^T W h - b^T h$$

induces a prior

$$p(h) \propto \exp(b^T h + \frac{1}{2} h^T W^T W h).$$

This prior is not motivated by the structure of any kind of data. It occurs as a consequence of designing the RBM to have a factorial posterior. The role of the deeper layers of the RBM is to provide a better prior on the first layer hidden units. The distribution over the first layer hidden units changes over time during learning, both because they become more adapted to the data and because the byproduct prior introduced by the RBM structure changes.

We propose to avoid this problem, at least at the second layer, by building on top of a model with a factorial prior. The second layer thus needs accomplish only one task–modeling correlations that are present in the data–and does not need to additionally work to drive out correlations that were induced by the first layer model.

We propose to use the S3C model for the first layer. Many first layer models are possible. For the reasons articulated above, we want a layer with an independent prior on its hidden units. A

directed model with binary v and h does not admit tractable mean field inference. Fortunately, a class of models with real-valued v allow tractable variational inference. Among this class of models, we choose S3C because we expect it is useful for modeling natural images. The related ssRBM (Courville *et al.*, 2011) is already known to be a successful generative model, so it is reasonable to expect that S3C would be a good generative model if equipped with a better prior.

If we assume that μ becomes large relative to α, then the primary structure we need to model is in h. We therefore propose placing a DBM prior rather than a factorial prior on h. The resulting model can be viewed as a deep Boltzmann machine with directed connections at the bottom layer. We call this model a *partially directed deep Boltzmann machine* (PD-DBM).

3.1 Model definition

Formally, the PD-DBM model consists of an observed input vector $v \in \mathbb{R}^D$, a vector of slab variables $s \in \mathbb{R}^{N_0}$, and a set of binary vectors $\boldsymbol{h} = \{h^{(0)}, \ldots, h^{(L)}\}$ where $h^{(l)} \in \{0,1\}^{N_l}$ and L is the number of layers added on top of the S3C model.

The model is parameterized by β, α, and μ, which play the same roles as in S3C. The parameters $W^{(l)}$ and $b^{(l)}$, $l \in \{0, \ldots, L\}$ provide the weights and biases of both the S3C model and the DBM prior attached to it. Following (Goodfellow *et al.*, 2012), we avoid overparameterizing the S3C distribution by restricting W to have unit norm, and restrict α to be a diagonal matrix and β to be a diagonal matrix or a scalar for computational and statistical efficiency.

Together, the complete model implements the following probability distribution:

$$P_{\mathrm{PD-DBM}}(v, s, \boldsymbol{h}) = P_{\mathrm{S3C}}(v, s | h^{(0)}) P_{\mathrm{DBM}}(\boldsymbol{h}).$$

A version of the model with three hidden layers ($L = 2$) is depicted graphically in Fig. 1.

Besides admitting a straightforward learning algorithm, the PD-DBM has several useful properties:

- The partition function exists for all parameter settings. This is not true of the ssRBM.
- The model family is a universal approximator. The DBM portion, which is a universal approximator of binary distributions (Le Roux and Bengio, 2008), can implement a one-hot prior on $h^{(0)}$, thus turning the overall model into a mixture of Gaussians, which is a universal approximator of real-valued distributions (Titterington *et al.*, 1985).
- Inference of the posterior involves feedforward, feedback, and lateral connections. This increases the biological plausibility of the model, and enables it to learn and exploit several rich kinds of interactions between features. The lateral interactions make the lower level features compete to explain the input, and the top-down influences help to obtain the correct representations of ambiguous input.

4 Learning in the PD-DBM

As with the DBM, maximum likelihood learning is intractable for the PD-DBM. Like S3C, it suffers from an intractable posterior distribution over the latent variables. Like the DBM, the PD-DBM additionally suffers from an intractable partition function.

Fortunately, the same approach used by Salakhutdinov and Hinton (2009) to train DBMs may be used to train the PD-DBM: rather than maximizing the log likelihood, we maximize a variational lower bound on the log likelihood. We do so using gradient ascent, making a sampling-based approximation to the gradient.

The basic strategy of variational learning is to approximate the true posterior $P(h, s \mid v)$ with a simpler distribution $Q(h, s)$. The choice of Q induces a lower bound on the log likelihood called the negative variational free energy. The term of the negative variational free energy that depends on the model parameters is

$$\begin{aligned} &\mathbb{E}_{s,h\sim Q}[\log P(v, s, h)] \\ =& -\mathbb{E}_{s,h\sim Q}[\log P(v \mid s, h^{(0)}) + \log P(s \mid h) + \log P(h)] \end{aligned}$$

For some models, changing to this objective function is sufficient to make learning tractable. In the case of the DBM and PD-DBM, the objective function is still not tractable because of the intractable DBM partition function. We can use contrastive divergence (Hinton, 2000) or stochastic maximum likelihood (Younes, 1998; Tieleman, 2008) to make a sampling-based approximation to the DBM partition function's contribution to the gradient.

The PD-DBM model has a nice property in that only a subset of the variables must be sampled during training. The factors of the partition function originating from the S3C portion of the model are still tractable; only those arising from the DBM must be approximated. In particular, training does not ever require sampling real-valued variables. This is a nice property because it means that the gradient estimates are bounded for fixed parameters and data. When sampling real-valued variables, it is possible for the sampling procedure to make gradient estimates arbitrarily large.

We found that using the "true gradient" (Douglas *et al.*, 1999) method to be useful for learning with the norm constraint on $W^{(0)}$. We also found that using momentum (Hinton, 2010) is very important when training jointly.

5 Inference procedure

The goal of variational inference is to maximize the lower bound on the log likelihood with respect to the approximate distribution Q over the unobserved variables. This is accomplished by selecting the Q that minimizes the Kullback–Leibler divergence:

$$\mathcal{D}_{KL}(Q(h,s)\|P(h,s|v)) \tag{2}$$

where $Q(h,s)$ is drawn from a restricted family of distributions. This family can be chosen to ensure that learning and inference with Q is tractable. We use the variational family

$$Q(s,h) = \Pi_{i=1}^{N_0} Q(s_i, h_i^{(0)}) \Pi_{l=1}^{L} \Pi_{i=1}^{N_l} Q(h_i^{(l)}).$$

Observing that eq. (2) is an instance of the Euler-Lagrange equation, we find that the solution must take the form

$$\begin{aligned} &Q(h_i^{(l)} = 1) = \hat{h}_i^{(l)}, \\ &Q(s_i \mid h_i^{(0)}) = \mathcal{N}(s_i \mid h_i^{(0)} \hat{s}_i,\ (\alpha_i + h_i W_i^T \beta W_i)^{-1}). \end{aligned} \tag{3}$$

where $\hat{h}_i$ and $\hat{s}_i$ must be found by an iterative process. In principle, nearly any optimization algorithm could be used to perform variational inference, but since inference is in the inner loop of learning, we require this optimization procedure to be fast. Inference in the S3C model is difficult, and the PD-DBM inherits this difficulty. We adopt the method of Goodfellow *et al.* (2012) for inferring $\hat{s}$ and modify it slightly to account top-down influence from $h^{(1)}$ when inferring $\hat{h}^{(0)}$. We use this method because it is design to run quickly on parallel architectures such as GPUs. The remaining DBM layers are comparatively easy to infer; their variational parameters can be updated with standard mean field fixed point equations as in (Salakhutdinov and Hinton, 2009). We describe the entire algorithm in detail in Algorithm 1).

Note that Algorithm 1 does not specify a convergence criterion. Many convergence criteria are possible–the convergence criterion could be based on the norm of the gradient of the KL divergence with respect to the variational parameters, the amount that the KL divergence has decreased in the last iteration, or the amount that the variational parameters have changed in the final iteration. Salakhutdinov and Hinton (2009) use the third approach when training deep Boltzmann machines and we find that it works well for the PD-DBM.

6 Sampling results

In order to demonstrate the improvements in the generative modeling capability conferred by adding a DBM prior on h, we trained an S3C model and a PD-DBM model on the MNIST dataset. We chose to use MNIST for these experiments because it is easy for a human observer to qualitatively judge whether samples come from the same distribution as this dataset.

Algorithm 1 `Fast Parallel Variational Inference Algorithm`

Let ρ be a user-defined hyperparemeter, with $\rho \in [0, 1]$. (We used $\rho = 0.5$)
$\forall l \in \{0, \ldots, L\}$ initialize $\hat{h}^{(l)}(0) = \sigma(b^{(l)})$.
Initialize $\hat{s}(0) = \mu$.
while not converged **do**
  Choose a layer to update (we cycle through updating all odd then all even layers, but other schedules are valid).
  if updating $\hat{s}$ **then**
    Compute the individually optimal value $\hat{s}_i^*$ for each i simultaneously:

$$\hat{s}_i^* = \frac{\mu_i \alpha_{ii} + v^T \beta W_i - W_i \beta \left[\sum_{j \neq i} W_j \hat{h}_j \hat{s}_j\right]}{\alpha_{ii} + W_i^T \beta W_i}$$

    Clip reflections by computing

$$c_i = \rho \text{sign}(\hat{s}_i^*) |\hat{s}_i|$$

    for all i such that $\text{sign}(\hat{s}_i^*) \neq \text{sign}(\hat{s}_i)$ and $|\hat{s}_i^*| > \rho |\hat{s}_i|$, and assigning $c_i = \hat{s}_i^*$ for all other i.
    Damp the updates by assigning

$$\hat{s}_i \leftarrow \eta_s c + (1 - \eta_s) \hat{s}$$

    where $\eta_s \in (0, 1]$, chosen by starting with a user-defined guess and backtracking if the KL divergence increases.
  else if updating $\hat{h}^{(0)}$ **then**
    Compute the individually optimal values for $\hat{h}^{(0)}$:

$$z_i = \left(v - \sum_{j \neq i} W_j \hat{s}_j \hat{h}_j - \frac{1}{2} W_i \hat{s}_i\right)^T \beta W_i \hat{s}_i + b_i - \frac{1}{2} \alpha_{ii} (\hat{s}_i - \mu_i)^2 - \frac{1}{2} \log(\alpha_{ii} + W_i^T \beta W_i) + \frac{1}{2} \log(\alpha_{ii})$$

$$\hat{h}_i^{(0)*} = \sigma(z_i + W^{(1)} \hat{h}^{(1)})$$

    Damp the update to $\hat{h}^{(0)}$:

$$\hat{h}^{(0)} \leftarrow \eta_h \hat{h}^{(0)*} + (1 - \eta_h) \hat{h}^{(0)}$$

    where $\eta_h \in (0, 1]$, chosen by starting with a user-defined guess and backtracking if the KL divergence increases.
  else if updating $\hat{h}^{(l)}$ **then**
    Assign

$$\hat{h}^{(l)} \leftarrow \sigma\left(b^{(l)} + \hat{h}^{(l-1)T} W^{(l)} + W^{(l+1)} \hat{h}^{(l+1)}\right)$$

    where the term for layer $l + 1$ is dropped if $l + 1 > L$.
  end if
end while

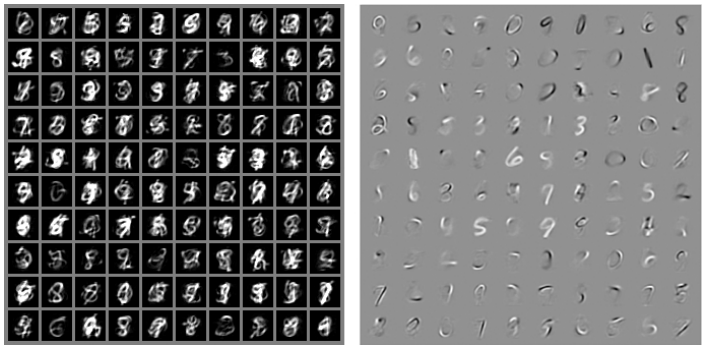

Figure 2: *Left:* Samples drawn from an S3C model trained on MNIST. *Right:* The filters used by this S3C model.

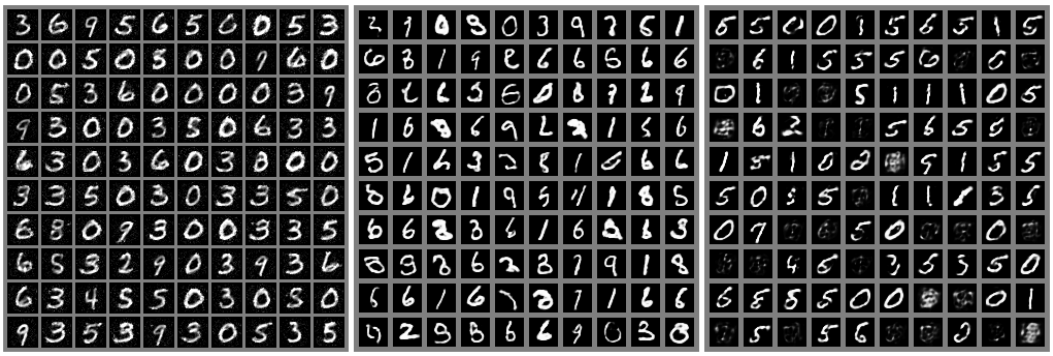

Figure 3: *Left:* Samples drawn from a PD-DBM model trained on MNIST using joint training only. *Center:* Samples drawn from a DBM model of the same size, trained using greedy layerwise pre-training followed by joint training. *Right:* Samples drawn from a DBM trained using joint training only.

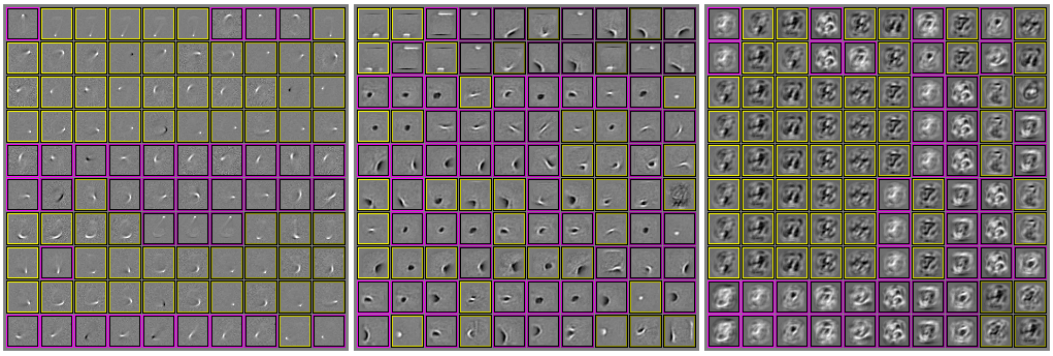

Figure 4: Each panel shows a visualization of the weights for a different model. Each row represents a different second layer hidden unit. We show ten units for each model corresponding to those with the largest weight vector norm. Within each row, we plot the weight vectors for the ten most strongly connected first layer units. Black corresponds to inhibition, white to excitation, and gray to zero weight. This figure is best viewed in color–units plotted with a yellow border have excitatory second layer weights while units plotted with a magenta border have inhibitory second layer weights. *Left:* PD-DBM model trained jointly. Note that each row contains many similar filters. This is how the second layer weights achieve invariance to some transformations such as image translation. This is one way that deep architectures are able to disentangle factors of variation. One can also see how the second layer helps implement the correct prior for the generative task. For example, the unit plotted in the first row excites filters used to draw 7s and inhibits filters used to draw 1s. Also, observe that the first layer filters are much more localized and contained fewer templates than those in Fig. 2 right. This suggests that joint training has a significant effect on the quality of the first layer weights; greedy pretraining would have attempted to solve the generative task with more templates due to S3C's independent prior. *Center:* DBM model with greedy pretraining followed by joint training. These weights show the same disentangling and invariance properties as those of the PD-DBM. Note that the filters have more black areas. This is because the RBM must use inhibitory weights to limit hidden unit activities, while S3C accomplishes the same purpose via the explaining-away effect. *Right:* DBM with joint training only. Note that many of the second layer weight vectors are duplicates of each other. This is because the second layer has a pathological tendency to focus on modeling a handful of first-layer units that learn interesting responses earliest in learning.

For the PD-DBM, we used $L = 1$, for a total of two hidden layers. We did not use greedy, layerwise pretraining– the entire model was learned jointly. Such joint learning without greedy pretraining has never been accomplished with similar deep models such as DBMs or DBNs.

The S3C samples and basis vectors are shown in Fig. 2. The samples do not resemble digits, suggesting that S3C has failed to model the data. However, inspection of the S3C filters shows that S3C has learned a good basis set for representing MNIST digits using digit templates, pen strokes, etc. It simply does not have the correct prior on these bases and as a result activates subsets of them that do not correspond to MNIST digits. The PD-DBM samples clearly resemble digits, as shown in Fig. 3. For comparison, Fig. 3 also shows samples from two DBMs. In all cases we display the expected value of the visible units given the hidden units. To more directly compare the differences between these models' parameters, we display a visualization of the weights of the models that shows how the layers interact in Fig. 4.

The first DBM was trained by running the demo code that accompanies (Salakhutdinov and Hinton, 2009). We used the same number of *units* in each layer in order to make these models comparable (500 in the first layer and 1,000 in the second). This means that the PD-DBM has a slightly greater number of *parameters* than the DBM, since the first layer units of the PD-DBM have both mean and precision parameters while the first layer units of the DBM have only a bias parameter. Note that the DBM operates on a binarized version of MNIST while S3C and the PD-DBM regard MNIST as real-valued. Additionally, the DBM demo code uses the MNIST labels during generative training while the PD-DBM and S3C were not trained with the benefit of the labels. The DBM demo code is hardcoded to pretrain the first layer for 100 epochs, the second layer for 200 epochs, and then jointly train the DBM for 300 epochs. We trained the PD-DBM starting from a random initialization for 350 epochs.

The second DBM was trained using two modifications from the demo code in order to train it in as similar a fashion to our PD-DBM model as possible: first, it was trained without access to labels, and second, it did not receive any pretraining. This model was trained for only 230 epochs because it had already converged to a bad local optimum by this time. This DBM is included to provide an example of how DBM training fails when greedy layerwise pretraining is not used. DBM training can fail in a variety of ways and no example should be considered representative of all of them.

7 Conclusion

We have introduced a deep generative model of real-valued data and learning and inference procedures for that model. We have demonstrated that the model can succesfully learn to generate samples from the MNIST dataset. The success of the jointly trained two layer version of this model provides some evidence for our hypothesis that DBMs may not be trained jointly because the difficulty of responding to the changing prior induced by the lower layer RBMs is too great.

In future work, we hope to quantify the PD-DBM's generative performance, evaluate its classification performance, and compare the performance of a greedily trained PD-DBM to the performance of a jointly trained PD-DBM.

References

Courville, A., Bergstra, J., and Bengio, Y. (2011). Unsupervised models of images by spike-and-slab RBMs. In *Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11)*.

Desjardins, G., Courville, A., and Bengio, Y. (2012). On training deep Boltzmann machines. Technical Report arXiv:1203.4416v1, Université de Montréal.

Douglas, S., Amari, S.-I., and Kung, S.-Y. (1999). On gradient adaptation with unit-norm constraints.

Garrigues, P. and Olshausen, B. (2008). Learning horizontal connections in a sparse coding model of natural images. In *NIPS'07*, pages 505–512. MIT Press, Cambridge, MA.

Goodfellow, I., Courville, A., and Bengio, Y. (2012). Large-scale feature learning with spike-and-slab sparse coding. In *Proc. ICML'2012*.

Hinton, G. E. (2000). Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Unit, University College London.

Hinton, G. E. (2010). A practical guide to training restricted Boltzmann machines. Technical Report UTML TR 2010-003, Department of Computer Science, University of Toronto.

Le Roux, N. and Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, **20**(6), 1631–1649.

Lücke, J. and Sheikh, A.-S. (2011). A closed-form EM algorithm for sparse coding. arXiv:1105.2493.

Mohamed, S., Heller, K., and Ghahramani, Z. (2012). Bayesian and l1 approaches to sparse unsupervised learning. In *ICML'2012*.

Montavon, G. and Müller, K.-R. (2012). Learning feature hierarchies with cented deep Boltzmann machines.

Salakhutdinov, R. and Hinton, G. (2009). Deep Boltzmann machines. In *Proc. AISTATS'2009*, volume 8.

Salakhutdinov, R. and Larochelle, H. (2010). Efficient learning of deep Boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), JMLR W&CP*, volume 9, pages 693–700.

Salakhutdinov, R., Tennenbaum, J., and Torralba, A. (2010). One-shot learning with a hierarchical nonparametric bayesian model. Technical report, MIT. MIT Technical Report MIT-CSAIL-TR-2010-052.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *ICML 2008*, pages 1064–1071. ACM.

Titsias, M. K. and Lázaro-Gredilla, M. (2011). Spike and slab variational inference for multi-task and multiple kernel learning. In *NIPS'2011*.

Titterington, D., Smith, A., and Makov, U. (1985). *Statistical Analysis of Finite Mixture Distributions*. Wiley, New York.

Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In *NIPS 17*, Cambridge, MA. MIT Press.

Younes, L. (1998). On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. In *Stochastics and Stochastics Models*, pages 177–228.

Zhou, M., Chen, H., Paisley, J. W., Ren, L., Sapiro, G., and Carin, L. (2009). Non-parametric Bayesian dictionary learning for sparse image representations. In *NIPS'09*, pages 2295–2303.