Sparse Maximum Entropy Deep Belief Nets

How Jing and Yu Tsao

Research Center for Information Technology Innovation, Academia Sinica, Republic of China

***Abstract*—In this paper, we present a sparse maximum entropy (SME) learning algorithm for deep belief net (DBN). The SME algorithm aims to maximize the entropy and encourage sparsity of the model. Compared with the conventional maximum likelihood (ML) learning, the proposed SME algorithm enables DBN to be more unbiased to data distributions and robust to over-fitting issues, and accordingly provide a better generalization capability. MNIST and NORB data sets were used to evaluated the proposed SME algorithm. Experimental results show that SME-trained DBN outperforms ML-trained DBN on both data sets.**

I. Introduction

LEARNING effective feature representations from unlabeled data has received considerable interest in recent years. Many unsupervised learning algorithms have been proposed to take advantage of large amounts of unlabeled data to learn a good feature representation [16], [28], [17]. Among these algorithms, deep belief network (DBN) is a popular method. DBN comprises multi-layer of feature detectors, where each detector modeled by an undirected graphical model is called restricted Boltzmann machine (RBM) [10]. To perform classification, a set of RBMs is first trained recursively in a layer-by-layer manner. Then, DBN is constructed by stacking up RBMs. Next, we can keep the parameters learned in DBN and place a logistic layer at the top, treating it as a feed-forward neural network. Finally, the parameters are fine-tuned using back-propagation algorithm to predict the correct class labels.

The most popular learning algorithm for RBM is the maximum likelihood (ML) learning, which aims to make the model "explain" the data well. Therefore when performing top-down inference, the model will generate samples resembling the observed training data. A ML-trained DBN (denoted hereinafter as ML-DBN) can then be constructed by stacking up these RBMs. Although ML-DBN provides satisfactory performance in many tasks, including, text [23], speech [18], and vision [13], the ML criterion may not be the optimal method for initializing the parameters in a deep neural network (DNN) for the subsequent discriminative fine-tuning. A good example is that deep Boltzmann machine (DBM), using a different algorithm to initialize the weights in DNN, provides superior performance over ML-DBN [24]. In addition, the ML learning often encounters over-fitting problems when the amount of training data is too limited.

This study proposes a sparse maximum entropy (SME) learning algorithm for RBM. The SME algorithm aims to maximize the entropy and encourage sparsity of the model. Compared with the conventional ML learning, the SME algorithm makes RBM more unbiased to unseen data and thus robust to possible over-fittings. The corresponding deep model, which we is called SME-DBN can then be constructed by stacking up SME-trained RBMs. Our experimental results on two image classification tasks demonstrate that DNN initialized by SME-DBN outperforms that initialized by ML-DBN, suggesting that SME-DBN provides a better starting point than the ML-DBN for the DNN classifier.

The remainder of this paper is organized as follows: section II and III introduce the preliminaries and backgrounds. Then, section IV shows the proposed algorithm and provides some justifications. Next, section V presents our experimental setup and evaluations of generative property and unsupervised feature learning, as well as classification performance. Finally, section VI concludes this paper.

II. Preliminaries

This section reviews related background knowledge of RBM and the ME learning algorithm.

A. Restricted Boltzmann Machine

A restricted Boltzmann machine (RBM) is an undirected bipartite graphical model with its probability distribution governed by an energy function E, parameterized by $\lambda = \{W, b, c\}$:

$$E(v,h;\lambda) = -\sum_{i\in visible} c_i v_i - \sum_{j\in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{1}$$

where v_i and h_j are the binary state of visible unit i and hidden unit j; c_i and b_j are their corresponding biases; w_{ij} is the weight between visible and hidden unit. The joint probability for configuration v, h is formulated as follows:

$$p(v,h;\lambda) = \frac{exp(-E(v,h;\lambda))}{Z}, \tag{2}$$

where Z is the partition function, which ensures the function a valid probability function. The conditional probability in this RBM is thus,

$$p(v_i|h;\lambda) = sigm(c_i + \sum_j w_{ij} h_j), \tag{3}$$

$$p(h_j|v;\lambda) = sigm(b_j + \sum_i w_{ij} v_i), \tag{4}$$

where $sigm$ is the sigmoid function. Parameter learning of this model is usually done by the ML learning. The log probability that the network assigns to the data is given by

$\log p(v;\lambda) = \log \sum_h p(v,h;\lambda)$. Taking the partial derivative of $\log p(v;\lambda)$ with respect to the parameters $\{W,c,b\}$ yields the following update rule:

$$W_{ij} = W_{ij} + < v_i h_j >_{data} - < v_i h_j >_{model}, \quad (5)$$

$$c_i = c_i + < v_i >_{data} - < v_i >_{model}, \quad (6)$$

$$b_j = b_j + < h_j >_{data} - < h_j >_{model}, \quad (7)$$

where $<>$ denotes the expectation. $<>_{data}$ means empirical expectation, and $<>_{model}$ denotes expectation with respect to model distribution.

This learning rule, i.e., Eqs. (5)-(7), has a nice interpretation: in order to do parameter learning, the probability that the network assigns to the observed data (also known as the positive phase) should be raised, while the probability of what the network believes (also known as the negative phase) should be reduced [1]. Therefore when learning is finished (the gradient vanishes), what network believes will be the same as the observed data. However, this learning algorithm is computationally intractable since to compute the expectation of the model, we need to enumerate exponentially many number of configurations. Simple Monte Carlo method using Gibbs sampling can be used to get the samples from this expectation, but it's very slow to wait for the Markov chain to mix. To address this problem, contrastive divergence (CD) method has been proposed to approximate the ML learning. In the case of CDk, "negative phase" samples from the model are drawn by performing alternating Gibbs sampling only k times instead of reaching the equilibrium [27], [7].

B. Gaussian Unit RBM

RBM using Eqs. (1)-(4) can only model binary random variable data. To model real-value data, the energy function should be modified in the following manner:

$$E(v,h;\lambda) = \frac{1}{2\sigma^2}\sum_i v_i^2 - \frac{1}{\sigma^2}(\sum_i c_i v_i + \sum_j b_j h_j + \sum_{i,j} v_i w_{ij} h_j), \quad (8)$$

and the conditional probabilities become

$$p(v_i|h;\lambda) = \mathcal{N}(c_i + \sum_j w_{ij}h_j, \sigma^2), \quad (9)$$

$$p(h_j|v;\lambda) = sigm(\frac{1}{\sigma^2}(b_j + \sum_i w_{ij}v_i), \quad (10)$$

where $\mathcal{N}(.)$ denotes the Gaussian distribution. Now the input data is modeled by a Gaussian, and hence this type of RBM is called Gaussian-Bernoulli RBM [11].

C. Deep Belief Network

Previous studies have shown that a single RBM has limited data modeling capability, whereas a deep model, formed by stacking up several RBMs, presents great data modeling power [2]. Hinton et al. proposed an efficient algorithm to train DBN by greedily training each layer of RBM using hidden activations in the previous layer [10]. It has been shown that the variational bound of the data log-likelihood is guaranteed under this greedy layer-wise learning framework, suggesting that stacking another layer of RBM will not deteriorate the model's generative power. Fig. 1 demonstrates the greedy layer-wise learning procedure. By stacking up RBMs, instead of getting a multi-layers Boltzmann machine, a hybrid model is constructed, where the top two layers form an RBM, and the lower layers form a directed belief net [2], [10]. This hybrid model is called deep belief network (DBN), with its probability function given by:

$$p(v,h_1,h_2,...,h_l) = p(h_{l-1},h_l)p(v|h_1)\prod_{2}^{i=l-1} p(h_{i-1}|h_i). \quad (11)$$

where l denotes the total number of layers. Fig. 2 demonstrates the hybrid structure of DBN.

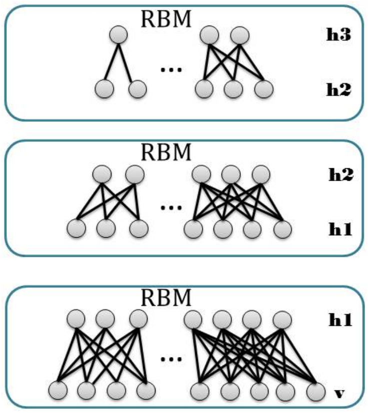

Fig. 1: Greedy layer-wise learning for DBN.

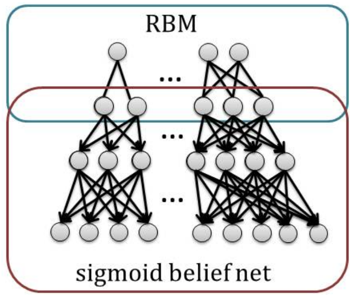

Fig. 2: Hybrid model of DBN after greedy layer-wise learning: top two layers form an RBM; bottom layers form a directed belief net.

D. Annealed Importance Sampling

Comparing generalization capabilities of different RBMs is quite difficult due to the intractable estimate of partition functions. Let $p_A(x) = \tilde{p_A}(x)/Z_A$ and $p_B(x) = \tilde{p_B}(x)/Z_B$, where $\tilde{p}$ denotes the unnormalized probability, importance sampling (IS) method formulates the ratio of partitions Z_A over Z_B as:

$$\begin{aligned}\frac{Z_A}{Z_B} &= \frac{\int \tilde{p_A}(x)}{Z_B}dx \\ &= \int \frac{\tilde{p_A}(x)}{\tilde{p_B}(x)} p_B dx \\ &\approx E_{p_B}[\frac{\tilde{p_A}(x)}{\tilde{p_B}(x)}]. \end{aligned} \tag{12}$$

The above ratio of partition functions can be computed by using a simple Monte Carlo approximation. However, if p_A and p_B are not close enough, this method may yield unreliable estimates [25]. To overcome this issue, annealed importance sampling (AIS) defines a sequence of intermediate distributions to estimate the importance weight [20],

$$\begin{aligned}\frac{\tilde{p_A}(x)}{\tilde{p_B}(x)} &= \frac{\tilde{p_1}(x)}{\tilde{p_B}(x)}\frac{\tilde{p_2}(x)}{\tilde{p_1}(x)}\frac{\tilde{p_3}(x)}{\tilde{p_2}(x)}\cdots\frac{\tilde{p_K}(x)}{\tilde{p_{K-1}}(x)}\frac{\tilde{p_A}(x)}{\tilde{p_K}(x)} \\ &= w^i_{AIS}, \end{aligned} \tag{13}$$

then the ratio of partition functions becomes

$$\frac{Z_A}{Z_B} = \frac{1}{M}\sum_{i=1}^{M} w^i_{AIS}. \tag{14}$$

A previous study has shown that AIS provides accurate estimate of the ratio of partition functions [25]. In addition, with a careful design of the partition function Z_B, AIS allows us to obtain an estimate of the partition function Z_A for any single RBM (by setting Z_B to be a constant we know of).

E. Estimating the Likelihood of DBN

The likelihood in the validation set is an important evaluation metric for generative models. However, it is rather difficult to estimate the exact likelihood for DBN. Nevertheless, we can obtain a lower bound on the likelihood by using Jensen's inequality:

$$\begin{aligned}\log p(v) &= \log \sum_{\mathbf{h}} p(v,\mathbf{h}) \\ &= \log \sum_{\mathbf{h}} Q(\mathbf{h}|v)\frac{p(v,\mathbf{h})}{Q(\mathbf{h}|\mathbf{v})} \\ &\geq \sum_{\mathbf{h}} Q(\mathbf{h}|v)\log\frac{p(v,\mathbf{h})}{Q(\mathbf{h}|\mathbf{v})} \\ &= \sum_{\mathbf{h}} Q(\mathbf{h}|v)\log p(v,\mathbf{h}) + H(Q(\mathbf{h}|v)) \\ &\approx E_Q[\log p(v,\mathbf{h}) - \log Q(\mathbf{h}|v)], \end{aligned} \tag{15}$$

where the inequality holds based on Jensen's inequality, and $\mathbf{h} = \{h_l, h_2, ...h_l\}$ denotes the entire set of hidden layers in DBN.

Now the Monte Carlo method can be used to estimate the lower bound of the log-likelihood, $\log p(v)$. First, we draw samples from the posterior distribution $Q(\mathbf{h}|v)$, which is available during the RBM training stages. Next, from Eq. (11) we can write $\log p(v,\mathbf{h}) = \log p(h_l, h_{l-1}) + \log p(v|h_1) + \sum_{i=2}^{l-1}\log p(h_{i-1}|h_i)$. To compute $\log p(v,\mathbf{h})$, the main difficulty is the partition function of the top-level RBM. We can again use AIS to estimate the partition function by letting $\log p(h_l, h_{l-1}) = \log \tilde{p}(h_l, h_{l-1}) - \log Z$, where the unnormalized probability, $\tilde{p}(h_l, h_{l-1})$, is easy to compute.

F. Maximum Entropy Learning

The ME learning aims to maximize the entropy of a model:

$$\max_p \; H(p) = -\int p\log p \tag{16}$$

subject to some constraints that prevent learning a naive model (a Gaussian distribution in the continuous case). This learning framework is widely used in natural language processing (NLP) where the input dimension for most tasks is the vocabulary size, which can easily exceed tens of thousands [5], [22], [21]. However, the amount of training data is usually insufficient to estimate a good probabilistic model. Thus, maximizing the entropy makes the model unbiased with respect to unseen data so it generalizes better.

III. Sparse Maximum Entropy Deep Belief Network

In this section, we introduce the proposed sparse maximum entropy deep belief network (SME-DBN).

A. Maximum Entropy Restricted Boltzmann Machine

Applying ME learning to RBM, the optimization problem is,

$$\max_p \; H(p) = -\sum_{v,h} p(v,h;\lambda)\log p(v,h;\lambda), \tag{17}$$

subject to

$$\sum_{v,h} p(v,h;\lambda)f(v) = \mu(\tilde{v}), \tag{18}$$

$$\sum_{v,h} p(v,h;\lambda)f(h) = \mu(h), \tag{19}$$

$$\sum_{v,h} p(v,h;\lambda)f(v,h) = \mu(\tilde{v},h), \tag{20}$$

where we have defined three types of features $f(v)$, $f(h)$, $f(v,h)$, each corresponding to the activated state of v, h, $v^T h$; $\mu(\tilde{v})$, $\mu(h)$, $\mu(\tilde{v},h)$ correspond to their empirical expectations, which are estimated from the data by

$$\mu(\tilde{v}) = \frac{\sum_{i=1}^N \tilde{v}_i}{N}, \tag{21}$$

$$\mu(h) = \frac{\sum_{i=1}^N \sum_h p(h|\tilde{v_i};\lambda)h}{N}, \tag{22}$$

$$\mu(\tilde{v},h) = \frac{\sum_{i=1}^N \sum_h p(h|\tilde{v_i};\lambda)\tilde{v_i}h^T}{N}. \tag{23}$$

A previous study shows that solving the constrained optimization problem (i.e., Eqs. (17)-(23)) is equivalent to solving another unconstrained optimization problem [30].

$$\max \; Q(\lambda,\lambda') = \sum_{v\in\tilde{X}}\sum_h p(h|v;\lambda')\log p(v,h;\lambda), \tag{24}$$

where the Q function is derived from the standard decomposition of the log-likelihood, $L(\lambda)$:

$$L(\lambda) = Q(\lambda, \lambda^{'}) + H(\lambda, \lambda^{'}), \tag{25}$$

where

$$\begin{aligned} Q(\lambda, \lambda^{'}) &= \sum_{v \in \tilde{X}} \sum_{h} p(h|v; \lambda^{'}) \log p(v, h; \lambda), \\ H(\lambda, \lambda^{'}) &= -\sum_{v \in \tilde{X}} \sum_{h} p(h|v; \lambda^{'}) \log p(h|v; \lambda). \end{aligned}$$

Thus, our problem can be solved using the EM algorithm. In the E-step, we compute the expectation for the three types of features as defined earlier. Next, we maximize the Q function in the M-step. Computing the expectation in the E-step is easy for RBM because inference is simple. However, maximizing the Q function in the M-step is unsurprisingly intractable, so approximate methods are needed.

The original estimation of Eqs (17)-(23) is not a convex optimization problem because there are latent variables in the model, whereas maximizing the Q function itself is a convex problem. Thus, we can apply either iterative scaling [4], [8] or generalized EM [6] along with Monte Carlo approximation to solve the problem. The overall learning procedure is demonstrated in **Algorithm1**.

Algorithm 1 EM-IS(CD) algorithm

```
Input: The parameter set in an RBM is denoted by $\lambda = \{W, c, b\}$ where $p(v, h; \lambda) = \frac{\exp(-E(v,h;\lambda))}{Z}$ and the
    energy function $E$ is defined in Eq. (1).
Initialize: Let $\lambda^{(0)} = W, c, b$ randomly initialized.

E-step: Given the current estimate of $\lambda^{(j)}$, compute the
    empirical expected value $\mu$ for all features given by
    Eqs. (21)-(23):

M-step: (maximizing $Q$ function) Run either IS or CD until
    $|\lambda^{new} - \lambda^{old}| < \epsilon$ .
 1: For IS : Let $f^{\#} = \sum_i f_i(v, h)$. Each update is given
    by:
    $\lambda_i^{new} = \lambda_i^{old} + \gamma_i$,
        $\gamma_i$ satisfies: $\sum_{v,h} f_i(v, h) e^{\gamma_i f^{\#}} p(v, h; \lambda^{(old)}) = \mu_i$
        (which is approximated by Gibbs sampling and
    Newton's method in our implementation).
 2: For CD: Perform gradient ascent with update rule of
    $\lambda_i$:
        $\frac{\partial Q}{\partial \lambda_i} = \mu_i - \sum_{v,h} p(v, h; \lambda^{(old)}) \frac{\partial E(v, h; \lambda^{(old)})}{\partial \lambda^{(old)}}$,
    where CD is applied to approximate the second expectation.
Repeat: until convergence.
Return: $\lambda^*$
```

B. Encouraging Sparsity

Sparse modeling shows its power in many domains. The proposed SME algorithm encourages sparsity to enforce each feature detector to learn independent useful features for explaining the data. The sparsity constraint is incorporated in the objective function to make the expected activation of each neuron small. In other words, the expected value $E[h_j|v]$ for each neuron to be turned on with respect to input data is small. Adding this constraint into our objective function makes the M-step of our algorithm as:

$$\begin{aligned} \max\ Q(\lambda, \lambda^{'}) = \sum_{\tilde{v} \in \tilde{X}} \sum_{h} p(h|\tilde{v}; \lambda) \log p(\tilde{v}, h; \lambda) \\ -\beta KL(\rho \| \hat{\rho_j}), \end{aligned} \tag{26}$$

where β is the parameter that controls the trade-off between sparsity and data fit, ρ is the sparsity target, and $\hat{\rho_j} = \frac{1}{|\tilde{X}|} \sum_{v \in \tilde{X}} E[h_j|v]$. $KL(\rho \| \hat{\rho_j})$ denotes the KL divergence between two Bernoulli distributions, where $KL(\rho \| \hat{\rho_j}) = \rho \log(\frac{\rho}{\hat{\rho_j}}) + (1 - \rho) \log(\frac{1-\rho}{1-\hat{\rho_j}})$.

C. Greedy Layer-Wise Learning

For standard ML-trained DBN, greedy layer-wise learning is justified by the variational lower bound on the log-likelihood, so adding another layer of RBM using previous hidden activations as new data will not reduce the total log-likelihood [2]. In SME-DBN, we can obtain a similar argument because the log-likelihood is also at the local optimum at the optimal point of the M-step. The difference between ME-DBN and ML-DBN is that among those local optima, DBN chooses the one with a higher likelihood, whereas SME-DBN chooses the one with a higher entropy [30]. Given that the log-likelihood is also locally optimal in SME-DBN, we can use the same argument to conclude that greedy layer-wise learning in SME-DBN still guarantees the variational lower bound on the data log-likelihood. Furthermore, SME maximizes the entropy, so higher entropy is obtained in SME-DBN compared to standard ML-DBN due to

$$\begin{aligned} H(v, h_1, h_2)^{ME} - H(v, h_1, h_2)^{ML} \\ \geq H(h_1, h_2)^{ME} - H(h_1, h_2)^{ML} \\ \geq 0. \end{aligned} \tag{27}$$

The inequality holds by the property of joint entropy ($H(a, b, c) >= H(a, b)$ for any random variable a, b, c).

IV. Experiment

The experiments in this study can be divided into two parts. First, we evaluated the generative property of SME-RBM and the capability of their feature learning. Second, we compared the classification performance of SME-DEN and ML-DBN on two image classification problems.

1) Data Sets and Protocol: Two benchmark data sets, i.e., **MNIST** and **NORB**, were selected for the evaluations. In Fig. 3, left and right panels show examples of the **MNIST** and **NORB** data, respectively. **MNIST** is a standard computer vision task that provides images containing 28×28 gray-scale pixels representing ten handwritten digits (0 to 9) [14]. The training and testing sets included 60,000 and 10,000 images, respectively. We retained 10,000 images from the training set as validation data. Preprocessing simply involved dividing each pixel by 255 to make it suitable for the RBM. The **NORB** data set is a 3D object recognition data set. In this study, we used a uniform-normalized set from [15] where the training set contained 24,300 pairs of stereo images, and the test set also contained 24,300 pairs of stereo images. The **NORB** data set covers five generic categories, including *cars*, *trucks*, *planes*, *animals*, and *humans*. Each image had 96×96 grayscale pixels values. Thus each training example contains 2×96×96 = 18,432 dimensions. To reduce the computation, we first ignored the outer part of the image and only kept the center 64×64 pixels (since the objects are placed mainly at the center). Then, we resized the image to 48×48 leading to 2×48×48 = 4,608 dimensions for each sample. Next, we followed the procedure as suggested in [29] to train a Gaussian-Bernoulli RBM with architecture 4,608-4,000 to convert the gray-scale images into binary variables. Finally, we treated these 4,000 variables as our preprocessed data that were fed into our algorithm.

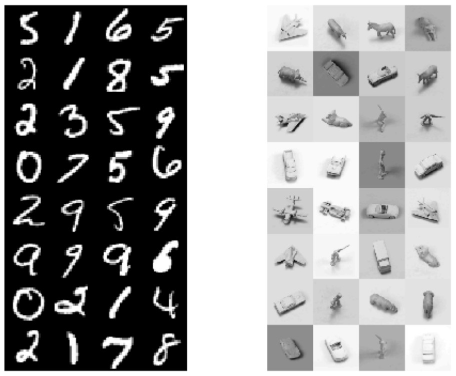

Fig. 3: Examples of **MNIST** (left) and **NORB** (right) data.

2) Generative Property Evaluation: The **MNIST** data set is used to conduct generative property evaluation. In this experiment, we constructed two toy models of RBM with architecture 784-10, one trained with ML and another trained with SME. We intentionally used a small number of hidden units so that the computation of the partition function was tractable. Both models were trained using the CD method (CDk, where k = 3). We compared the log-likelihood and entropy with varying amounts of data for training. Table 1 shows the exact log-likelihood for the test set and the joint entropy given by SME-RBM and ML-RBM.

The toy models was probably not sufficiently powerful to explain the data well because it had only 10 hidden units. Therefore, we also conducted an experiment using RBM with architecture 784-100, where the partition function was no longer tractable; in this case, we used AIS to approximate the partition function. Table 2 shows the estimated likelihood and entropy with architecture 784-100. We ran AIS for 500 steps and used 15,000 intermediate distributions for AIS. The entropy was estimated by running Monte Carlo method for 100,000 steps.

	Log-likelihood		Joint Entropy	
# data	ML	SME	ML	SME
1000	-177.0023	-175.9402	0.0934	0.6791
10000	-170.4505	-161.6452	0.0223	0.9944
60000	-158.8122	-158.1970	0.5940	2.8907

TABLE I: Exact log-likelihood on test set and entropy with varying amounts of training data for ML-RBM and SME-RBM (both with architecture 784-10).

	Log-likelihood		Joint Entropy	
# data	ML	SME	ML	SME
1000	-160.0999	-161.2354	118.7784	131.7914
10000	-126.7226	-127.5376	123.8056	143.7566
60000	-124.6619	-128.4414	138.7706	168.4937

TABLE II: Estimated log-likelihood lower bound on test set and entropy with varying amounts of training data for ML-RBM and SME-RBM (both with architecture 784-100).

Tables 1 and 2 show that the proposed SME-RBM always had a much higher entropy than ML-RBM with either training data size= 1,000, 10,000, or 60,000. We also found that the SME-learning algorithm did not sacrifice much likelihood as the entropy gain increased.

3) Unsupervised Feature Learning: The second experiment visualized RBM trained using the SME and ML learning algorithms. This set of experiments was conducted using the **MNIST** data set. We used RBM with architecture 784-500. Figs. 4 and 5 illustrate the ML and SME learned RBMs, respectively. The two figures demonstrate that our method learned a clearer edge detectors, which may come from the sparsity penalty that enables each neuron to learn a clearer representation of the input image. More classification evaluations will be conducted to compare the two models, but it is confirmed here that the two RBMs learned different feature representations.

4) Classification Performance: We tested classification performances on both **MNIST** and **NORB** data sets. For both tasks, after learning the features without using any information from the labels, we retained these weights and stacked another logistic layer at the top, treating it as a deterministic feed-forward neural network. We then used standard back-propagation to train the neural network. For the **MNIST** data set, we constructed the DBN with architecture 784-500-500-2000 and the sparsity target 0.1. For the **NORB** data set, we construct the DBN with architecture 4000-2000-2000 and the sparsity target 0.1. The upper and lower rows in Table 3 list the classification error rates (in $\%$) for the **MNIST** and **NORB** test sets, respectively. For **MNIST**, SME-DBN gives 1.05% error rate, and this is compared to 1.2% by ML-DBN [12], 1.4% by SVM [9] and 1.6% by MLP with randomly initialized weights [26]. For **NORB**, our method achieves 10.63% classification error rate, which outperforms ML-DBN (11.9%) [19], and SVM (11.6%) [3].

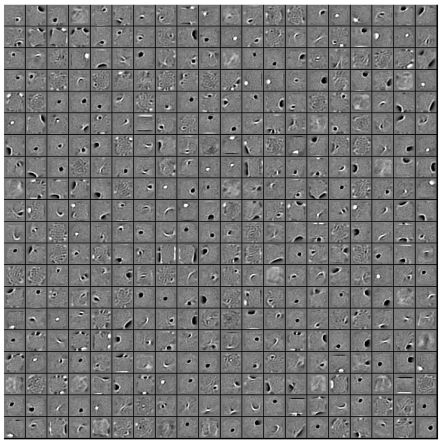

Fig. 4: Feature detectors learned by 784-500 ML-RBM.

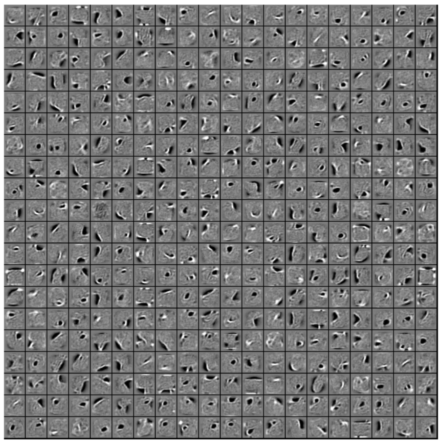

Fig. 5: Feature detectors learned by 784-500 SME-RBM.

Data Set	SME-DBN	ML-DBN	SVM	MLP
MNIST	**1.05**%	1.2%	1.4%	1.6%
NORB	**10.63**%	11.9%	11.6%	–

TABLE III: Classification error rates (in%).

V. Conclusion

Recently, unsupervised feature learning has attracted much attention because it can take advantage of large amounts of unlabeled data. In this paper, we present an SME learning algorithm for DBN. The proposed SME learning algorithm maximizes entropy and promote sparsity of the model. Experimental results on MNIST and NORB data sets confirm that the SME-trained DBN generalizes better than the ML-trained DBN.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. "A learning algorithm for boltzmann machines". *Cognitive Science*, pages 147–169, 1985.

[2] Y. Bengio. *"Learning Deep Architectures for AI"*. Hanover, MA, USA, 2009.

[3] Y. Bengio and Y. Lecun. *"Scaling learning algorithms towards AI"*. MIT Press, 2007.

[4] A. Berger. "The improved iterative scaling algorithm: A gentle introduction". 1997.

[5] A. Berger, S. D. Pietra, and V. D. Pietra. "A maximum entropy approach to natural language processing". *Computational Linguistics*, 22:39–71, 1996.

[6] J. A. Bilmes. "A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models". Technical report, 1997.

[7] M. A. Carreira-Perpinan and G. E. Hinton. "On contrastive divergence learning".

[8] J. N. Darroch and D. Ratcliff. "Generalized iterative scaling for Log-Linear models". *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[9] D. Decoste and B. Schölkopf. "Training invariant support vector machines". *Mach. Learn.*, 46(1-3):161–190, March 2002.

[10] G. E. Hinton, S. Osindero, and Y. W. Teh. "A fast learning algorithm for deep belief nets". *Neural Comput.*, 18(7):1527–1554, 2006.

[11] G.E. Hinton. "A practical guide to training restricted boltzmann machines". Technical report, 2010.

[12] G.E. Hinton and R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". *Science*, 313(5786):504–507, July 2006.

[13] A. Krizhevsky. "Convolutional deep belief networks on cifar-10", 2010.

[14] Y. Lecun and C. Cortes. "The MNIST database of handwritten digits".

[15] Y. LeCun, F. J. Huang, and L. Bottou. "Learning methods for generic object recognition with invariance to pose and lighting". In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'04, pages 97–104, Washington, DC, USA, 2004. IEEE Computer Society.

[16] H. Lee, A. Battle, R. Raina, and A. Y. Ng. "Efficient sparse coding algorithms". In *In NIPS*, pages 801–808. NIPS, 2007.

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations". In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[18] A. Mohamed, G. E. Dahl, and G. E. Hinton. "Acoustic modeling using deep belief networks". *Trans. Audio, Speech and Lang. Proc.*, 20(1):14–22, January 2012.

[19] V. Nair and G.E. Hinton. "3-d object recognition with deep belief nets". In *Advances in Neural Information Processing Systems 22*, 2009.

[20] R. M. Neal. "Annealed importance sampling". *Statistics and Computing*, 11(2):125–139, April 2001.

[21] K. Nigam. "Using maximum entropy for text classification". In *In IJCAI Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[22] A. Ratnaparkhi. *"Maximum entropy models for natural language ambiguity resolution"*. PhD thesis, 1998.

[23] R. Salakhutdinov and G. E. Hinton. "Semantic hashing". *Int. J. Approx. Reasoning*, 50(7):969–978, July 2009.

[24] R. Salakhutdinov and G.E. Hinton. "Deep Boltzmann machines'. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

[25] R. Salakhutdinov and I. Murray. "On the quantitative analysis of deep belief networks".

[26] P. Y. Simard, D. Steinkraus, and J. C. Platt. "J.C.: Best practices for convolutional neural networks applied to visual document analysis". pages 958–963, 2003.

[27] I. Sutskever and T. Tieleman. "On the convergence properties of contrastive divergence". In *Proc. Conference on AI and Statistics (AI-Stats)*, 2010.

[28] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[29] N. Vinod and G. E. Hinton. "Implicit mixtures of restricted boltzmann machines".

[30] S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans. "The latent maximum entropy principle".