A sparse-response deep belief network based on rate distortion theory

Nan-Nan Ji, Jiang-She Zhang*, Chun-Xia Zhang

School of Mathematics and Statistics, Xi’an Jiaotong University, Xi’an Shaanxi 710049, China

Abstract

Deep belief networks (DBNs) are currently the dominant technique for modeling the architectural depth of brain, and can be trained efficiently in a greedy layer-wise unsupervised learning manner. However, DBNs without a narrow hidden bottleneck typically produce redundant, continuous-valued codes and unstructured weight patterns. Taking inspiration from rate distortion (RD) theory, which encodes original data using as few bits as possible, we introduce in this paper a variant of DBN, referred to as sparse-response DBN (SR-DBN). In this approach, Kullback–Leibler divergence between the distribution of data and the equilibrium distribution defined by the building block of DBN is considered as a distortion function, and the sparse response regularization induced by L1-norm of codes is used to achieve a small code rate. Several experiments by extracting features from different scale image datasets show that our approach SR-DBN learns codes with small rate, extracts features at multiple levels of abstraction mimicking computations in the cortical hierarchy, and obtains more discriminative representation than PCA and several basic algorithms of DBNs.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Recent neuroscience findings [1–6] have provided insight into the principles governing information representation in the mammal brain, leading to new ideas for designing systems to effectively represent information. One of the key findings is that the mammal brain is organized in a deep architecture. Given an input percept, it is represented with multiple levels of abstraction, each level corresponding to a different area of cortex. By using these abstract features learned through the deep architecture, human achieves perfect performance on many real-world tasks such as object recognition, detection, prediction, and visualization. In current machine learning community, how to imitate this hierarchical architecture of mammal brain to obtain good representation of data in order to improve the performance of a learning algorithm has become an essential issue.

Recently, deep learning has become the dominant technique to learn good information representation that exhibits similar characteristics to that of the mammal brain. It has gained significant interest for building hierarchical representations from unlabeled data. A deep architecture consists of feature detector units arranged in multiple layers: lower layers detect simple features and feed into higher layers, which in turn detect more complex features. In particular, deep belief network (DBN), the most popular approach of deep learning, is a multilayer generative model in which each layer encodes statistical dependencies among the units in the layer below, and it can be trained to maximize (approximately) the likelihood of its training data. So far, there have been a great deal of DBN models being proposed. For example, Hinton et al. [7] proposed an algorithm based on learning individual layers of a hierarchical probabilistic graphical model from the bottom up. Bengio et al. [8] proposed a similar greedy algorithm on the basis of auto-encoders. Ranzato et al. [9] developed an energy-based hierarchical algorithm, using a sequence of sparsified auto-encoders/decoders. Particularly, the model proposed by Hinton et al. is a breakthrough for training deep networks. It can be viewed as a composition of simple learning modules, each of which is a restricted Boltzmann machine (RBM) that contains a layer of visible units representing observable data and a layer of hidden units learned to represent features that capture higher-order correlations in the data [10]. Nowadays, DBNs have been successfully applied to a variety of real-world applications, including hand-written character recognition [7,11], text representation [12], audio event classification, object recognition [13], human motion capture data [14,15], information retrieval [16], machine transliteration [17], speech recognition [18–20] and various visual data analysis tasks [21–23].

Although DBNs have demonstrated promising results in learning good codes or representations, DBNs without constraints on the hidden layers may produce redundant, continuous-valued codes and unstructured weight patterns. Some scholars attempted to further improve DBNs’ performance according to add constraints...
on the representations [24–30]. Among them, introducing the notion of sparsity makes DBN present state-of-the-art results because sparse representation is able to obtain succinct codes and structured weight patterns. Given images, sparse DBN is able to discover low-level structures such as edges, as well as high-level structures such as corners, local curvatures, and shapes.

Sparsity was first proposed as a model of simple cells in the visual cortex [31]. Up to now, it has been a key element of DBNs learning through exploiting a variant of auto-encoders, RBM, or other unsupervised learning algorithms. In practice, there are many ways to enforce some form of sparsity on the hidden layer representation of a deep architecture. The first successful deep architecture exploiting sparsity of representation involved auto-encoders [9]. Sparsity was achieved with a so-called sparsifying logistic, by which the codes are obtained with a nearly saturating logistic whose offset is adapted to maintain a low average number of times that the code is significantly non-zero. One year later the same group introduced a somewhat simple variant [28] through assigning a Student-t prior on the coders. In the past, the Student-t prior was used to obtain sparse MAP estimates for the codes generating an input [29] in computational neuroscience models of the V1 visual cortex area. Another approach also related to computational neuroscience involves two levels of sparse RBMs [24]. Sparsity is achieved with a regularization term that penalizes a deviation of the expected activation of hidden units from a fixed low-level. One level of sparse coding of images results in filters very similar to those seen in V1. When training a sparse deep belief network, the second level appears to detect visual features similar to those observed in area V2 of visual cortex.

From the point of view of information theory, one of the major principles for finding concise representations is rate distortion (RD) theory [32], which focuses on the problem of determining minimal amount of information that should be communicated over a channel, so that a compressed representation of the original data can be approximately reconstructed at the output data without exceeding a given distortion. Sparse coding methods can be interpreted as special cases of RD theory [33]. For deep multi-layer neural networks, hidden layers without narrow bottleneck may result in redundant and continuous-valued codes [34]. We hold that incorporating the constraint of a minimum rate of information flow into the training process of multi-layer neural networks is able to make networks obtain succinct representations. From this point of view, we propose in this paper a novel version of sparse DBNs for unsupervised feature extraction by taking inspiration from the idea of RD theory. In DBNs, activation probability of the hidden units over a data vector is always fixed, that is, \(\mathbb{E} = \frac{1}{Z} \exp(-E(v, h))\),

\[ Z = \sum_{v} \exp(-E(v, h)), \]

where \(Z\) is a normalization constant. \(E(v, h)\) denotes the energy of the state \((v, h)\). If the visible units are binary-valued, the energy function can be defined as

\[ E(v, h) = -\sum_{i=1}^{D} \sum_{j=1}^{K} v_i w_{ij} h_j - \sum_{j=1}^{K} b_j h_j - \sum_{i=1}^{D} c_i v_i, \]

where \(b_j\) and \(c_i\) are respectively hidden and visible unit biases. If the visible units are real-valued, we can define the energy function by adding a quadratic term to make the distribution well defined, that is,

\[ E(v, h) = -\frac{1}{2} \sum_{i=1}^{D} v_i^2 - \sum_{i=1}^{D} \sum_{j=1}^{K} v_i w_{ij} h_j - \sum_{j=1}^{K} b_j h_j - \sum_{i=1}^{D} c_i v_i. \]

From the energy function, we can see that the hidden units \(h_j\) are independent of each other when conditioning on \(v\) since there are no direct connections between hidden units. Similarly, the visible units \(v_i\) are also independent of each other when conditioning on \(h\).

2. Deep belief network (DBN) and its building block

DBNs are probabilistic generative models that contain many layers of hidden variables, in which each layer captures high-order-correlations between the activities of hidden features in the layer below. A key feature of this algorithm is its greedy layer-by-layer training that can be repeated several times to learn a deep, hierarchical model. The main building block of a DBN is a bipartite undirected graphical model called the Restricted Boltzmann Machine (RBM). In this section, we provide a brief technical overview of RBM and the greedy learning algorithm for DBNs.

2.1. Restricted Boltzmann machine (RBM)

RBM [10, 11, 35, 36] is a two-layer, bipartite, undirected graphical model with a set of (binary or real-valued) visible units (random variables) \(v\) of dimension \(D\) representing observable data, and a set of binary hidden units (random variables) \(h\) of dimension \(K\) learned to represent features that capture higher-order correlations in the observable data. These two layers are connected by a symmetrical weight matrix \(W \in \mathbb{R}^{D \times K}\), whereas there are no connections within a layer. Fig. 1 illustrates the undirected graphical model of an RBM.

RBM can be viewed as a Markov random field that tries to represent input data with hidden units. Here, the weights encode a statistical relationship between the hidden units and the visible units. The joint distribution over the visible and hidden units is defined by

\[ P(v, h) = \frac{1}{Z} \exp(-E(v, h)), \]

where \(Z\) is a normalization constant. \(E(v, h)\) denotes the energy of the state \((v, h)\). If the visible units are binary-valued, the energy function can be defined as

\[ E(v, h) = -\sum_{i=1}^{D} \sum_{j=1}^{K} v_i w_{ij} h_j - \sum_{j=1}^{K} b_j h_j - \sum_{i=1}^{D} c_i v_i, \]

where \(b_j\) and \(c_i\) are respectively hidden and visible unit biases. If the visible units are real-valued, we can define the energy function by adding a quadratic term to make the distribution well defined, that is,

\[ E(v, h) = -\frac{1}{2} \sum_{i=1}^{D} v_i^2 - \sum_{i=1}^{D} \sum_{j=1}^{K} v_i w_{ij} h_j - \sum_{j=1}^{K} b_j h_j - \sum_{i=1}^{D} c_i v_i. \]
Specifically, the units of a binary hidden layer, conditioned on the visible layer, are independent Bernoulli random variables. The binary state, $h_j$, of each hidden unit, $j$, is set to 1 with probability

$$p(h_j = 1|v) = \sigma \left( \sum_i W_{ij}v_i + b_j \right).$$  \hfill (5)$$

where $\sigma(s) = 1/(1 + \exp(-s))$ is the sigmoid function. If the visible units own binary values, the visible units, conditioned on the hidden layer, are also independent Bernoulli random variables. In this case, the binary state, $v_i$, of each visible unit, $i$, is set to 1 with probability

$$p(v_i = 1|h) = \sigma \left( \sum_j W_{ij}h_j + c_i \right).$$  \hfill (6)$$

If the visible units own real values, the visible units, conditioned on the hidden layer, are independent Gaussian random variables as follows, namely,

$$p(v_i|h) = \mathcal{N} \left( \sum_j W_{ij}h_j + c_i, 1 \right).$$  \hfill (7)$$

where $\mathcal{N}(\cdot, \cdot)$ stands for a Gaussian distribution.

Since RBM is a generative model, its parameters can be optimized by performing stochastic gradient descent on the log-likelihood of training data. The probability that the network assigns to a visible vector (training vector) is given by summing over all possible hidden vectors, i.e.,

$$p(v) = \sum_h \exp \left( -E(v, h) \right).$$  \hfill (8)$$

The derivative of the log probability of a training vector with respect to a weight is simple and it can be expressed as

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{\theta} - \langle v_i \rangle_{\theta} \langle h_j \rangle_{\theta},$$  \hfill (9)$$

where the angle brackets denote expectations under the distribution specified by the subscript that follows. $p^\theta$ is the distribution of the data, and $\theta^\theta$ is the equilibrium distribution defined by RBM. $\theta = \{W, b, c\}$ includes all parameters needed to learn. We can get a very simple learning rule of a weight $w_{ij}$ as

$$\Delta w_{ij} = \epsilon \left( \langle v_i h_j \rangle_{\theta} - \langle v_i \rangle_{\theta} \langle h_j \rangle_{\theta} \right).$$  \hfill (10)$$

where $\epsilon$ is a learning rate. Unfortunately, computing the exact gradient of the log probability of training data is intractable since getting an unbiased sample of $\langle v_i h_j \rangle_{\theta}$ is much more difficult. In general, this can be solved by starting at any random state of the visible units and performing alternating Gibbs sampling for a very long time. However, this method is time-consuming. Fortunately, Hinton [11] developed a much faster learning procedure by following an approximation to the gradient of a different objective function, called the “Contrastive Divergence” (CD). The new learning rule is then given by

$$\Delta w_{ij} = \epsilon \left( \langle v_i h_j \rangle_{\theta} - \langle v_i h_j \rangle_{\theta^\theta} \right)$$  \hfill (11)$$

where $p_{\theta^\theta}$ represents a distribution defined by running alternating Gibbs sampling, initialized at the data, for 1 step.

Similar to the learning rule of $w_{ij}$, the hidden and visible unit biases ($b$, and $c$) can be updated as

$$\Delta b_j = \epsilon \left( \langle h_j \rangle_{\theta} - \langle h_j \rangle_{\theta^\theta} \right),$$  \hfill (12)$$

$$\Delta c_i = \epsilon \left( \langle v_i \rangle_{\theta} - \langle v_i \rangle_{\theta^\theta} \right).$$  \hfill (13)$$

As for more detailed introduction of RBM, interested readers can refer to [37,38].

2.2. Deep belief network (DBN)

Several layers of RBMs can be used to compose a DBN which is a generative model consisting of many layers of hidden features. Specifically, each layer comprises a set of binary or real-valued units which capture high-order correlations between the activities of hidden features in the layer below. Two adjacent layers compose a RBM. Fig. 2 shows a DBN with two hidden layers $h^1$ and $h^2$. $v$ is the input layer, corresponding to the visible layer of the bottom RBM. Hinton et al. [7] proposed a fast, unsupervised learning algorithm for these deep networks in 2006. A key feature of this algorithm is its greedy layer-by-layer training that can be repeated several times to learn a deep, hierarchical model.

First, we train the bottom RBM with CD on the training data. The corresponding parameters $W_{ij}$, $b_j$, and $c_i$ will be frozen and the hidden unit values will be inferred. These inferred values then serve as the input data to train the next higher layer in the network to model the hidden layer representations of the first level RBM. This process can be repeated to yield a deep architecture that is an unsupervised model of the training distribution.

3. DBN based on RD theory

Generally speaking, the success of Machine Learning (ML) relies on a good feature representation of data, which is able to faithfully mimic certain properties of visual areas, leading to make progress towards Artificial Intelligence (AI). Research on the primate brain reveals that visual system processes information through multiple stages of transformation and representation [1–4]. Due to this evidence, learning high-level features using deep architecture becomes a major focus of ML research, and is getting more and more attention in recent years. DBNs, providing efficient learning algorithms for deep multi-layer neural networks, are currently the

![Fig. 2. Deep belief network with two hidden layers $h^1$ and $h^2$, left figure is RBM for $v$, and the right figure is RBM for $h^1$.](image)
dominant technique for learning high-level features efficiently and effectively.

However, DBNs without constraints on the hidden layers may produce redundant, continuous-valued codes and unstructured weight patterns. From this point of view, a large number of theories and methods in ML research have been developed for finding succinct codes and structured weight patterns [9,24,28,39–44]. One of the major principles in information theory for finding succinct codes is rate distortion (RD) theory [32] which explores effective coding of the original data, that is, an encoding scheme using as few bits as possible while maintaining the similarity between the original data and the decoded data within a certain allowable distortion level.

In this section, based on the idea of RD theory, we try to incorporate the learning of representation with a small code rate into the training of DBNs by adding constraints on each hidden layer. Because DBN can be trained efficiently by a greedy layerwise unsupervised learning procedure, i.e., one hidden layer is trained at a time by a RBM, incorporating constraints into the hidden layers of DBN can be carried out by adding constraint on the hidden layer of each stacked RBM.

3.1. Rate distortion theory

In information theory, the entropy of the random variable \( Z \) is defined as

\[
H(Z) = -\sum_{z} p(z) \log p(z),
\]

where \( p(z) \) is the probability distribution of \( Z \). The rate distortion function \( D \) is defined as

\[
D = \inf_{\hat{Z} \sim D_p} E[D(\hat{Z}, Z)],
\]

where \( D_p \) is the distortion measure, \( \hat{Z} \) is a random variable representing the rate of compression, i.e., code rate, and the optimization problem with respect to the conditional probability \( p(Z|V) \) is

\[
\min_{p(Z|V)} I(V,Z) \text{ s.t. } E[d(V,Z)] \leq D,
\]

where \( I(V,Z) \) denotes the mutual information between \( V \) and \( Z \), representing the rate of compression, i.e., code rate, and the expectation \( E[d(V,Z)] \) is taken with respect to the joint distribution \( p(V,Z) \). The dual formulation of this problem is

\[
\min_{p(V,Z)} E[d(V,Z)] \text{ s.t. } I(V,Z) \leq T,
\]

which is to find a coding scheme within a given rate such that the expected distortion between the original data and the data decoded from \( Z \) is minimized.

3.2. Sparse-response RBM based on RD theory

Note that the mutual information \( I(V,Z) \) measures the information shared by the variables \( V \) and \( Z \), and it can be calculated as

\[
I(V,Z) = H(Z) - H(Z|V),
\]

where \( H(Z) \) stands for the information entropy of the random variable \( Z \), and \( H(Z|V) \) denotes the conditional entropy. Furthermore, if we consider a deterministic coding scheme \( V \rightarrow Z \), then \( I(V,Z) = H(Z) - H(Z|V) = H(Z) \). As for RBMs, activation probability of the hidden units for a training vector is always regarded as its representation or code. Therefore, the code scheme obtained by RBM is deterministic. So, in our approach, we have \( I(V,Z) = H(Z) \), that is to say, minimum mutual information \( I(V,Z) \) in the RD theory can be transformed into the minimization of entropy \( H(Z) \).

In information theory, the entropy of the random variable \( Z \), \( H(Z) \), is a measure of the uncertainty in \( Z \). If some constraints are imposed on \( Z \), then the uncertainty of this variable is reduced, and the information entropy decreases. Based on the above analysis, we will reduce the information entropy of the random variable \( Z \) by a constraint, \( E[|Z|_1] \leq \eta \), that is, sparse representation is used to achieve a small code rate. Concretely, \( L_1 \)-regularization term on the activation probability of hidden units \( i.e., |p(h=1|\psi)\) \) is exploited to reduce the information entropy of representations learned by RBM. In this way, the sparseness of representation, \( E[|Z|_1] \leq \eta \), corresponds to the small rate \( I(V,Z) \leq T \).

Distortion \( d(V,Z) \) in the RD theory is considered as an error measure for representing the data \( V \) by the data decoded from \( Z \). In most cases, this distortion measure is defined as \( d(V,Z) = ||V - BZ||^2 \), where \( B \) is usually searched so that the reconstruction error is minimized. In our approach, because RBM is a probabilistic model and can be viewed as a Markov random field, we will use the Kullback–Leibler divergence, \( KL(p^0 || p_Z^0) \), between the distribution of the data, \( p^0 \), and the equilibrium distribution defined by a RBM, \( p_Z^0 \), to measure the error for representing the original data \( V \) by the decoded data from the activation probability of hidden units \( P(h=1|\psi) \).

As described above, we consider the Kullback–Leibler divergence between the original data’s distribution and the equilibrium distribution defined by RBM as a distortion function, and use the \( L_1 \)-norm of the activation probability of hidden units to achieve a small code rate. In doing so, the dual formulation of RD theory corresponds to the following problem:

\[
\min_{\{w_j, c_j, b_j\}} KL(p^0 || p_Z^0) \text{ s.t. } \sum_{i=1}^m \|p(h=0|\psi_i)||1 \leq \eta, \tag{16}
\]

which can be expressed in a trade-off form

\[
\min_{\{w_j, c_j, b_j\}} KL(p^0 || p_Z^0) + \lambda \sum_{i=1}^m \|p(h=0|\psi_i)||1, \tag{17}
\]

where \( \lambda \) is a regularization parameter that reflects the relative importance of the distortion with respect to the code rate. And, \( m \) represents the size of training data. This new problem will be referred to as SR-RBM henceforth.

In principle, we can apply gradient descent to solve this problem. However, computing the gradient of the Kullback–Leibler divergence, \( KL(p^0 || p_Z^0) \), is difficult because of the time required to approach equilibrium. Fortunately, it is much more effective to use the approximate gradient of the contrastive divergence, \( p^{(t)} || p_Z^{(t)} - p_Z^0 \), to replace the gradient of the Kullback–Leibler divergence, \( KL(p^0 || p_Z^0) \) [11]. For an RBM, this approximate gradient is particularly easy to compute:

\[
-\frac{\partial}{\partial w_j} (p^{(t)} || p_Z^{(t)} - p_Z^0) \approx \langle v_j h_i \rangle_{p^0} - \langle v_j h_i \rangle_{p_Z^{(t)}}. \tag{18}
\]

The detailed explanation for Eq. (18) can be found in [11]. In this way, we can employ a CD learning algorithm (introduced in Section 2) to approximate the gradient of the Kullback–Leibler divergence, \( KL(p^0 || p_Z^0) \), and then apply gradient descent to the regularization term, which is similar to the method used in [24]. This means that in each iteration we can apply the CD update rule, followed by one step of gradient descent using the gradient of the regularization term, as summarized in Algorithm 1. Specifically, the gradient of the regularization term over the parameters is as follows:

\[
-\frac{\partial}{\partial v_j} \sum_{i=1}^m \|p(h=0|\psi_i)||1 = -\sum_{i=1}^m \frac{\partial}{\partial v_j} \text{sigmoid} \left( b_j + \sum_{l=1}^d w_{jl} v_l^{(t)} \right) = -\sum_{i=1}^m \left( p_i^{(t)} (1-p_i^{(t)}) v_i^{(t)} \right). \tag{19}
\]
Fig. 3. Activation probability of hidden units for one natural image patch obtained by RBM (left) and SR-RBM (right).

Fig. 4. Visualization of some bases learned from the Olshausen natural image dataset. (a) 144 bases extracted by RBM; (b) 144 bases extracted by SR-RBM; and (c) 100 second layer bases learned by SR-DBN. Here, each second layer base was visualized as a weighted linear combination of the first layer bases.
3.3. Learning deep belief network using SR-RBM

In order to increase computational efficiency, in the gradient step that tries to minimize the regularization term, we update only the bias terms $b_j$'s (which directly control the degree to which the hidden units are activated), instead of updating all the parameters $b_j$ and $w_{ij}$'s.

**Algorithm 1.** SR-RBM learning algorithm.

1. Update the parameters using CD learning rule as follows:
   \[
   W_{ij} = W_{ij} + \epsilon \langle \langle V^l \rangle^b_b - \langle \langle V^l \rangle^h_b \rangle^h_b \rangle^h_b,
   \]
   \[
   c_i = c_i + \epsilon \langle \langle V^l \rangle^b_i - \langle \langle V^l \rangle^h_i \rangle^h_i \rangle^h_i,
   \]
   \[
   b_j = b_j + \epsilon \langle \langle h^l_j \rangle^b_j - \langle \langle h^l_j \rangle^h_j \rangle^h_j \rangle^h_j,
   \]

   where $\epsilon$ is a learning rate, and $\langle \langle \cdot \rangle \rangle^h_b$ is an expectation over the reconstruction data, estimated using one iteration of Gibbs sampling;

2. Update the parameters using the gradient of the regularization term by Eqs. (19) and (20);

3. Repeat Steps 1 and 2 until converge.

4. Experiments

In this section, we carry out some experiments on four image stimulus datasets (namely, natural images [32], MNIST dataset [45], NORB dataset [46] and CIFAR-10 dataset [47]) to evaluate and compare the performance of our algorithm with that of several other algorithms (i.e., PCA [48], RBM [10,11], DBN composed of RBM [7], SparseRBM [24] and SparseDBN [24]) qualitatively and quantitatively. In all experiments, we initialized weights and biases as some random numbers which come from a uniform distribution. To speed-up the learning process, we divided each dataset into mini-batches, and updated the weights after each mini-batch. Moreover, we preprocessed each image patch by subtracting the mean pixel value, and dividing the result by its standard deviation.

4.1. Natural image

We first tested our model’s ability to learn hierarchical representations of natural images. Training data are a set of $12 \times 12$ pixel natural image patches taken from ten $512 \times 512$ images of natural surroundings in the American northwest, made available by Olshausen et. al [32]. We used 100,000 $12 \times 12$ patches randomly sampled from these images, skipping over any patch within 1 pixel of the border of the image. Each subset of 200 patches was used as a mini-batch.

First, we trained SR-RBM and RBM models with 144 visible units and 144 hidden units. Fig. 3 shows activation probability of hidden units caused by one input data, i.e., representations of this input data obtained by RBM and SR-RBM. For other training data, representations obtained by these two models are similar to those illustrated in Fig. 3. From this figure, we can see that information entropy of the representation obtained by SR-RBM is much smaller than that obtained by RBM, because the values of most components of representation learnt by SR-RBM approximate to zeros (while the components corresponding to RBM spread over the whole interval $(0,1)$), which reduces uncertainty of the representations obtained by SR-RBM over all training patterns.

Fig. 4(b) shows the learned bases corresponding to SR-RBM. It can be observed that they are oriented, gabor-like bases and

![Fig. 5. Visualization of 196 bases learned by RBM (left) and SR-RBM (right) algorithms from the MNIST digit dataset.](image-url)
resemble the receptive fields of V1 simple cells. These results are consistent with much previous work [3,9,24,32,49,50]. In contrast, the standard RBM results in bases that were neither oriented nor localized, as shown in Fig. 4(a).

Then we learned a SR-DBN with two hidden layers by stacking one SR-RBM on top of another, and 144 and 50 units were used in the first and second hidden layers, respectively. The learned second layer bases were visualized as weighted linear combination of the first layer bases, and were illustrated in Fig. 4(c). Note that this visualization method will also be used in next experiments. From Fig. 4(c), we can see that many bases responded selectively to contours, corners, angles, and surface boundaries in the images. These results are qualitatively consistent with those reported in [24,51,52].

4.2. MNIST digit dataset

The MNIST digit dataset contains 60,000 training and 10,000 test images of 10 handwritten digits (0–9), each image with size 28 x 28 pixels [45]. In our experiments, to speed up the learning process, we sampled the first 2000 images per class to train our algorithm as well as other unsupervised feature extraction algorithms. The size of mini-batch was set to 200.

In order to qualitatively compare the bases obtained by SR-RBM and RBM, we first trained them with 784 visible units and 196 hidden units. The learning rates for weights and biases were set to 0.001. Although 196 is less than the input dimensionality of 784, the representation is still overcomplete because the effective dimension of the digit dataset is considerably less than 784 as pointed out in [9]. Fig. 5 shows the resulted bases learnt by SR-RBM and RBM. It can be found that the majority of the bases learnt by RBM appear to be local blob detectors, with only a few that specialize as little stroke detectors. Meanwhile, there were a few bases remaining uninformative (i.e., almost uniform random grey patches). With SR-RBM, a much larger proportion of interesting (visibly not random, blob detectors and with a clear structure) feature detectors were learnt. These include local oriented stroke detectors and detectors of digit parts such as loops. This result is consistent with those obtained by applying different algorithms to learn representations of this dataset [9,53]. The representations of

### Table 1

<table>
<thead>
<tr>
<th>p</th>
<th>100 Samples</th>
<th>500 Samples</th>
<th>1000 Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training errors</td>
<td>Testing errors</td>
<td>Training errors</td>
</tr>
<tr>
<td>0.005</td>
<td>0.07</td>
<td>10.34</td>
<td>2.86</td>
</tr>
<tr>
<td>0.01</td>
<td>0.06</td>
<td>9.98</td>
<td>2.80</td>
</tr>
<tr>
<td>0.02</td>
<td>0.05</td>
<td>9.17</td>
<td>2.68</td>
</tr>
<tr>
<td>0.03</td>
<td>0.05</td>
<td>9.44</td>
<td>2.69</td>
</tr>
</tbody>
</table>

### Table 2

<table>
<thead>
<tr>
<th>λ</th>
<th>100 Samples</th>
<th>500 Samples</th>
<th>1000 Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training errors</td>
<td>Testing errors</td>
<td>Training errors</td>
</tr>
<tr>
<td>0.02</td>
<td>0.04</td>
<td>9.37</td>
<td>3.04</td>
</tr>
<tr>
<td>0.03</td>
<td>0.04</td>
<td>8.32</td>
<td>2.51</td>
</tr>
<tr>
<td>0.04</td>
<td>0.05</td>
<td>8.24</td>
<td>2.47</td>
</tr>
<tr>
<td>0.05</td>
<td>0.06</td>
<td>8.24</td>
<td>2.84</td>
</tr>
</tbody>
</table>

![Fig. 6.](image_url) Activation probability of hidden units for one input pattern obtained by RBM (left) and SR-RBM (right) over MNIST digit dataset.

![Fig. 7.](image_url) Visualization of 50 bases of second hidden layer learned by DBN (left) and SR-DBN (right) algorithms from the MNIST digit dataset.
one input pattern learnt by RBM and SR-RBM models are shown in Fig. 6.

Furthermore, in order to learn hierarchical features, we then trained a second RBM and SR-RBM based on the activation probability of hidden units produced by RBM and SR-RBM in the above experiment. These stacked RBM and SR-RBM composed a DBN and a SR-DBN. Here, training was performed on 50 hidden units and we decreased the code rate (increase the value of $\lambda$) for SR-DBN. The bases generated by the second hidden layer are visualized in Fig. 7. These results indicate that the bases of the second hidden layer learnt by SR-DBN are more abstract than those obtained by DBN. It is obvious that they learned the digits distinctly. By contrast, the bases learnt by DBN remain uninteresting since they are almost uniform random grey patches. Therefore, SR-DBN is able to capture higher-order correlations among the input pixel intensities.

### Table 3
Error rate on MNIST training (with 100, 500 and 1000 samples per class) and test sets produced by a linear classifier trained on raw data and transformed codes produced by PCA, RBM, SparseRBM and SR-RBM.

<table>
<thead>
<tr>
<th></th>
<th>100 Samples</th>
<th>500 Samples</th>
<th>1000 Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training errors</td>
<td>Testing errors</td>
<td>Training errors</td>
</tr>
<tr>
<td>RAW</td>
<td>37.60</td>
<td>40.25</td>
<td>37.90</td>
</tr>
<tr>
<td>PCA</td>
<td>0.62</td>
<td>29.04</td>
<td>9.41</td>
</tr>
<tr>
<td>RBM</td>
<td>0.14</td>
<td>12.07</td>
<td>3.64</td>
</tr>
<tr>
<td>SparseRBM</td>
<td>0.05</td>
<td>9.17</td>
<td>2.68</td>
</tr>
<tr>
<td>SR-RBM</td>
<td>0.05</td>
<td>8.24</td>
<td>2.47</td>
</tr>
</tbody>
</table>

### Table 4
Error rate on MNIST test set produced by a classifier trained on the first and second layer activations produced by DBN, SparseDBN and SR-DBN. The first and second layer activations are concatenated as feature vectors fed into a classifier.

<table>
<thead>
<tr>
<th>Number of training samples per class</th>
<th>DBN</th>
<th>SparseDBN</th>
<th>SR-DBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>9.76</td>
<td>9.29</td>
<td>7.91</td>
</tr>
<tr>
<td>500</td>
<td>5.53</td>
<td>5.29</td>
<td>4.03</td>
</tr>
<tr>
<td>1000</td>
<td>4.36</td>
<td>4.35</td>
<td>3.85</td>
</tr>
</tbody>
</table>

Fig. 8. Error rate on MNIST training (with 100, 500 and 1000 samples per class) and test sets produced by a linear classifier trained on 100, 200, 300 and 400 transformed codes produced by SR-RBM, SparseRBM and PCA from the MNIST digit dataset.
To empirically verify the advantage of the representation learnt by SR-RBM in terms of its discriminative power, we first executed SR-RBM and several other unsupervised feature extraction algorithms (i.e., PCA, RBM and SparseRBM) on 20,000 images (i.e., 2000 images per class), then used their created representations as input for the same linear classifier. The number of hidden units for RBM, SparseRBM and SR-RBM was set to 500, and the number of principal components for PCA was also set to 500. For the classification process, we used representations of 100, 500 and 1000 images per class to train a linear classifier. The representations of remaining images were used for test. For each combination of training data size and algorithm, we trained 50 classifiers with randomly chosen training sets and then used the average classification error to evaluate the performance of the corresponding algorithm. Note that SparseRBM and SR-RBM involve some hyper-parameter (i.e., $\lambda$ and $p$ for SparseRBM, $\lambda$ for SR-RBM), the detailed algorithm of SparseRBM can be found in [24]), we took several values of these hyper-parameters to conduct experiments.

Table 1 displays the training and test error rates of a linear classifier when providing the representations generated by a SparseRBM as its input. Table 2 reports the results for SR-RBM. In order to facilitate the comparison, the best results were highlighted in bold. When compared with PCA and RBM, we chose the best results of SparseRBM and SR-RBM under each training sample size. In addition, the recognition ability based on raw data was also considered. These comparison results were displayed in Table 3. From Table 3, we can see that SR-RBM always achieves the best recognition accuracy on the training and test sets when using a different number of training samples. Furthermore, in order to compare the representations learnt by DBN, SparseDBN and SR-DBN in terms of their discriminative power, we trained RBM, SparseRBM and SR-RBM with 100 hidden units again, by considering the representations learnt by RBM, SparseRBM and SR-RBM mentioned above as visible data. We constructed representation for each image by concatenating the first and second layer activations, and trained the same classifier using these representations. Table 4 displays the error rate on the test set.

Moreover, recognition ability of hidden representations when throwing out those units whose activation times were lower than a threshold was also tested. Specifically, we trained linear classifiers on the activation probabilities of 100, 200, 300 and 400 hidden units among 500 units. These hidden units were activated by the most input images. For each algorithm (i.e., SR-RBM, SparseRBM and PCA), Fig. 8 plots the training and test error rates of a linear classifier as a function of the number of hidden units. From this figure, SR-RBM is seen to learn more discriminative hidden representations than SparseRBM and PCA. This advantage is more noticeable for small training datasets.

4.3. NORB dataset

NORB [46] is a considerably more difficult dataset than MNIST, which contains images of 50 different 3D toy objects with 10 objects in each of five generic classes: cars, trucks, planes, animals and humans. Each object is captured from different viewpoints and under various lighting conditions. The training set contains 24,300 stereo image pairs of 25 objects, 5 per class, while the test set contains 24,300 stereo pairs of the remaining, different 25 objects. Considering the computational cost, we subsampled the original $2 \times 108 \times 108$ stereo-pair images to $2 \times 32 \times 32$, and randomly sampled 10,000 images from the training set to train our algorithm and other unsupervised feature extraction algorithms. The size of mini-batch was set to 100. Some random samples used in our experiments are shown in Fig. 9.

First, we trained RBM and SR-RBM with 1000 hidden units. Fig. 10 demonstrates some of the extracted bases. It can be observed

<table>
<thead>
<tr>
<th>$p$</th>
<th>200 Samples</th>
<th>500 Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training errors</td>
<td>Testing errors</td>
</tr>
<tr>
<td>0.02</td>
<td>0.00</td>
<td>4.48</td>
</tr>
<tr>
<td>0.04</td>
<td>0.00</td>
<td>4.05</td>
</tr>
<tr>
<td>0.06</td>
<td>0.00</td>
<td>3.98</td>
</tr>
<tr>
<td>0.08</td>
<td>0.00</td>
<td>3.63</td>
</tr>
</tbody>
</table>

Table 5. Error rate on NORB training (with 200 and 500 samples per class) and test sets produced by a linear classifier trained on transformed codes produced by SparseRBM with several values of $p$. |  

Fig. 10. Visualization of 200 bases learned respectively by RBM (left) and SR-RBM (right) algorithms from the NORB dataset.
that most bases obtained by SR-RBM appear to be object-part detectors and contour detectors. As for RBM, the majority of the bases appear to be local blob detectors and edge detectors, with only a few that specialized as contour detectors. Therefore, SR-RBM is able to learn features that are more attractive.

In order to test the discriminative power of the representations learnt by SR-RBM, it was compared with representations learnt by RBM, SparseRBM and PCA. Similar to the experiment done with the MNIST dataset, we chose 200 and 500 samples per class as training set size. For each combination of training data size and algorithm, we trained 50 classifiers with randomly chosen training sets and then used the average classification error to evaluate the performance of the corresponding algorithm. Table 5 displays the training and test error rates of a linear classifier when providing the representations generated by a SparseRBM as its input. Table 6 reports the training and test error rates of a linear classifier when providing the representations generated by a SparseRBM as its input. Table 7 shows the averaged classification results obtained by training a linear classifier based on the activation probability of 500, 600, 700, 800 and 900 hidden units learnt by SparseRBM and SR-RBM. These hidden units were activated by the most input images. Meanwhile, classification results of 500, 600, 700, 800 and 900 principal components learnt by PCA are also displayed. From this figure, SR-RBM is seen to learn much more discriminative representations than SparseRBM and PCA.

**Table 6**

<table>
<thead>
<tr>
<th>λ</th>
<th>200 Samples</th>
<th>500 Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training errors</td>
<td>Testing errors</td>
</tr>
<tr>
<td>0.02</td>
<td>0.00</td>
<td>3.32</td>
</tr>
<tr>
<td>0.03</td>
<td>0.00</td>
<td>3.59</td>
</tr>
<tr>
<td>0.04</td>
<td>0.00</td>
<td>3.44</td>
</tr>
<tr>
<td>0.05</td>
<td>0.00</td>
<td>3.34</td>
</tr>
</tbody>
</table>

**Table 7**

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>200 Samples</th>
<th>500 Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training errors</td>
<td>Testing errors</td>
</tr>
<tr>
<td>PCA</td>
<td>0.02</td>
<td>9.67</td>
</tr>
<tr>
<td>RBM</td>
<td>0.00</td>
<td>4.95</td>
</tr>
<tr>
<td>SparseRBM</td>
<td>0.00</td>
<td>3.62</td>
</tr>
<tr>
<td>SR-RBM</td>
<td>0.00</td>
<td>3.32</td>
</tr>
</tbody>
</table>

**Table 8**

<table>
<thead>
<tr>
<th>Number of training samples per class</th>
<th>DBN</th>
<th>SparseDBN</th>
<th>SR-DBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>4.05</td>
<td>3.28</td>
<td>2.13</td>
</tr>
<tr>
<td>500</td>
<td>1.20</td>
<td>1.08</td>
<td>0.91</td>
</tr>
</tbody>
</table>

**Fig. 11**

Error rate on NORB training (with 200 and 500 samples per class) and test sets produced by a linear classifier trained on transformed codes produced by SR-RBM, SparseRBM and PCA.
4.4. CIFAR-10 dataset

The CIFAR-10 dataset consists of 32 × 32 color images of animals and vehicles. It contains 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Each class contains exactly 6000 images with 5000 train and 1000 test examples. In our experiment, we randomly sampled 20,000 images from the training (2000 images per class) to train our algorithm and other unsupervised feature extraction algorithms. The size of mini-batch was set to 200. Some random samples from the training set used in our experiment are shown in Fig. 12.

SR-DBN, DBN and SparseDBN composed by two stacked SR-RBM, two stacked RBM and two stacked SparseRBM were trained, where the number of units in the first and second hidden layers was respectively set to 1000 and 500. Fig. 13 shows some bases in the first hidden layer learnt by DBN and SR-DBN, and some bases learned in the second hidden layer are shown in Fig. 14. The first layer bases obtained by SR-DBN are oriented, gabor-like bases and resemble the receptive fields of V1 simple cells, which is similar to SparseDBN. In contrast, DBN results in bases that were neither oriented nor localized (almost uniform random pixels). Furthermore, we can observe that the second layer bases learnt by SR-DBN selectively responded to contours, corners, angles and surface boundaries in the images.

Similar to previous experiments, the discriminative power of representations learnt by SR-RBM was tested by a linear classifier which was trained on the activation probability of hidden units. The training set size was set to 1000. Experiments were made over 50 randomly chosen training sets. Tables 9 and 10 display the classification errors of a classifier when providing the representations obtained by SparseRBM and SR-RBM as its input. We chose the best results for comparison. Results are shown in Table 11. From this table, SR-RBM is seen to achieve the best recognition accuracy on the training and test sets.

5. Conclusions

We have presented in this paper a variant of DBN model for unsupervised feature extraction based on the idea of RD theory. This novel DBN (SR-DBN) is carried out by stacking several novel RBMs (SR-RBMs). In a SR-RBM, sparse response induced by a $L_1$ regularization term on the activation probability of hidden units is used to achieve small rate of codes. At the same time, the Kullback-Leibler divergence between original data’s distribution and the equilibrium distribution defined by the model is employed as a distortion function. As for the learning of SR-RBM, it can be easily trained by CD and gradient descend method. Experiments on four image datasets of different scales are carried out to confirm the effectiveness and superiority of our proposed new model. Compared with regular DBN, our new model SR-DBN learns local oriented, gabor-like edge filters in the first layer and the second layer unit capture contours, corners, angles, and surface boundaries in the images. Meanwhile, our results on image recognition tasks show that SR-RBM and SR-DBN learn more appropriate features for supervised tasks than RBM, DBN, SparseRBM, SparseDBN and PCA algorithms. One limitation of this algorithm is the determination of learning rate and parameter about regularizer terms which are carried out by try-and-error for all
experiments. How to properly select them is deserved to be further studied.

Conflict of interest

None declared.

Acknowledgments

This work was supported by the National Basic Research Program of China (973 Program) under Grant no. 2013CB329404, the National Natural Science Foundation of China under Grant nos. 91230101, 61075006, 11131006, 11201367, the Research Fund for the Doctoral Program of Higher Education of China under Grant nos. 20100201120048.

References

Nan-Nan Ji received the B.S. degree and the M.S. degree in Applied Mathematics from Chong'an University, Xi'an, China. She is currently a Ph.D. candidate with School of Mathematics and Statistics, Xi'an Jiaotong University. Her research interests are focused on neural network, deep learning, pattern recognition, and intelligence computation.

Jiang-She Zhang was born in 1962, and received his M.S. and Ph.D. degrees in Applied Mathematics from Xi'an Jiaotong University, Xi'an, China, in 1987 and 1993 respectively. He joined Xi'an Jiaotong University, China, in 1987, where he is currently a full Professor in School of Mathematics and Statistics. Up to now, he has authored and coauthored on monograph and over 50 journal papers on robust clustering, optimization, and short-term load forecasting for electric power system. His current research focus is on Bayesian learning, global optimization, ensemble learning, and deep learning.

Chun-Xia Zhang was born in 1980 and received her B.S. degree in Mathematics from Xinyang Normal University, Xinyang, China, in 2002. In 2005 and 2010, she received her M.S. and Ph.D. degrees respectively in Probability and Statistics, Applied Mathematics from Xian Jiaotong University, Xian, China. Currently, she is a Lecturer in School of Mathematics and Statistics at Xian Jiaotong University, China. She has published about 20 journal papers on ensemble learning techniques, and nonparametric regression. Her current research interests mainly include classifier combination strategies, bootstrap methods and deep learning.