Knowl Inf Syst (2017) 53:533–550
DOI 10.1007/s10115-017-1049-x

REGULAR PAPER

Effective sparsity control in deep belief networks using normal regularization term

Mohammad Ali Keyvanrad[1] · **Mohammad Mehdi Homayounpour**[1]

Received: 5 March 2016 / Revised: 10 March 2017 / Accepted: 29 March 2017 / Published online: 10 April 2017

Abstract Nowadays the use of deep network architectures has become widespread in machine learning. Deep belief networks (DBNs) have deep network architectures to create a powerful generative model using training data. Deep belief networks can be used in classification and feature learning. A DBN can be learned unsupervised, and then the learned features are suitable for a simple classifier (like a linear classifier) with a few labeled data. In addition, according to researches, by using sparsity in DBNs we can learn useful low-level feature representations for unlabeled data. In sparse representation, we have the property that learned features can be interpreted, i.e., correspond to meaningful aspects of the input, and capture factors of variation in the data. Different methods are proposed to build sparse DBNs. In this paper, we proposed a new method that has different behavior according to deviation of the activation of the hidden units from a (low) fixed value. In addition, our proposed regularization term has a variance parameter that can control the force degree of sparseness. According to the results, our new method achieves the best recognition accuracy on the test sets in different datasets with different applications (image, speech and text) and we can achieve incredible results when using a different number of training samples, especially when we have a few samples for training.

Keywords Deep belief network · Restricted Boltzmann machine · Normal sparse RBM · Quadratic sparse RBM · Rate distortion sparse RBM

An earlier version of this work was presented at IJCNN2015 [10].

✉ Mohammad Mehdi Homayounpour
homayoun@aut.ac.ir

Mohammad Ali Keyvanrad
keyvanrad@aut.ac.ir

1 Laboratory for Intelligent Multimedia Processing (LIMP), Amirkabir University of Technology, Tehran, Iran

1 Introduction

From many years ago, artificial neural networks have been used in artificial intelligence applications. Pattern recognition, voice and speech analysis and natural language processing are some of these applications that use artificial neural networks. Due to some theoretical and biological reasons, deep models and deep network architectures with many nonlinear processing layers were suggested.

These deep models have many layers and parameters that must be learnt. When the learning process is so complicated and a huge number of parameters are needed, artificial neural networks are rarely used. The problem of this number of layers is that training is time-consuming and training becomes trapped at local minima. Therefore, we cannot achieve acceptable results. One important tool for dealing with this problem is to use DBNs (deep belief networks) that can create neural networks, including many hidden layers [16] with unsupervised learning.

Deep belief networks can be used in classification and feature learning. Data representation is very important in machine learning. Therefore, much work has been done for feature preprocessing, feature extraction and feature learning. In feature learning, we can create a feature extraction system and then use the extracted features in classification and other applications. Using unlabeled data in high-level feature extraction [15] and also increasing discrimination between extracted features are the benefits of DBN for feature learning [1,8]. According to this advantage, DBN can be learnt unsupervised and then the learnt features are suitable for a simple classifier (like a linear classifier) with a few labeled data.

Layers of DBN are created from the restricted Boltzmann machine (RBM) that is a generative and undirected probabilistic model. RBMs use a hidden layer to model the probability distribution of visible variables. Indeed, we can create a DBN for hierarchical processing using stacking RBMs. Therefore, most of the improvements in DBNs are due to the improvement in RBMs. This paper studies an improvement in training by using a new regularization term in the sparse RBM model.

Sparsity has recently become a concept of great interest and has become a key ingredient in deep belief networks. Bengio has argued that if one is going to have fixed-size representations, then sparse representations are more efficient (than non-sparse ones) in an information-theoretical sense, allowing for variation in the effective number of bits per example [2].

Another argument in favor of sparsity is that the fixed-length representation is going to be used as input for further processing, so that it should be easy to interpret. A highly compressed encoding is usually highly entangled, so that no subset of bits in the code can really be interpreted unless all the other bits are taken into account. Instead, we would like our fixed-length sparse representation to have the property that individual bits or small subsets of these bits can be interpreted, i.e., correspond to meaningful aspects of the input, and capture factors of variation in the data [2].

According to researches, sparse coding learns useful low-level feature representations for unlabeled data [22]. Now, suppose that we are interested in learning more complex, higher level features. For example, we may want to learn object-part features from images rather than just edge detectors. With this goal in mind, it appears to be fairly plausible to attempt to build hierarchical feature representations using sparse coding. However, it is not straightforward to apply sparse coding recursively to build multiple levels of hierarchy. We hypothesize that this difficulty may be due to the following factors.

First, sparse coding (with real-valued inputs) assumes a non-sparse input distribution, whereas the output distribution of sparse coding is very sparse; applying another sparse coding on the top of the sparse coding output may not satisfy the modeling assumption. Second, the optimization (i.e., inference) in sparse coding is fairly expensive since it involves L_1 regularized optimization [14].

Therefore, an alternative approach has been used in building hierarchical representation. The main idea is to use deep learning algorithms that can provide hierarchical representation. One of these deep learning algorithms is DBN. Unlike sparse coding, the RBMs can be easily stacked to form a hierarchical representation called DBNs. Furthermore, approximate inference for RBMs and DBNs is computationally efficient [14]. In this paper, we develop a new sparse variant of the deep belief networks.

Different methods are proposed to build sparse RBMs [9,14,21]. These different methods are based on a different regularization term definition. In one of these definitions, the regularization term quadratically penalizes a deviation of the expected activation of the hidden units from a (low) fixed level p [14]. In another paper, based on the rate distortion theory, the penalty factor is the activation probability in hidden units [9]. In this paper, we proposed a new regularization term that has different behavior according to deviation of the activation of the hidden units from a (low) fixed level p and in addition, our proposed regularization term has a variance parameter that can control the force degree of sparseness. According to the results, our new method achieves the best recognition accuracy on the test sets in different datasets with different applications.

The rest of this paper is organized as follows: In Sect. 2, RBM and DBN are described. Sparse RBM methods and our proposed method in this paper are presented in Sects. 3 and 4. In Sect. 5, some experiments are conducted and the proposed method is compared to some other methods in the tasks of digit recognition on the MNIST dataset, spoken letter recognition on the ISOLET dataset and document topic classification on the 20 Newsgroups dataset. Finally, Sect. 6 concludes the paper.

2 Deep belief networks (DBNs) and restricted Boltzmann machines (RBMs)

DBNs are composed of multiple layers of RBMs. Outputs of hidden layer in each RBM can be considered as input for the next RBM layer (see Fig. 1). With this method, DBN will be trained layer by layer [7].

An RBM is a Markov random field with two groups of hidden and visible units. In RBMs, each neuron is connected to all neurons in the other layer. However, there are no connections between neurons in the same layer [4]. This restriction in RBMs causes conditional independence between visible and hidden units.

The joint probability distribution under the model uses an energy function of {v,h}.

$$\begin{aligned} E(\boldsymbol{v}, \boldsymbol{h}) &= -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h} - \boldsymbol{a}^T \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{h} \\ &= -\sum_{i=1}^{g_v} \sum_{j=1}^{g_h} w_{ij} v_i h_j - \sum_{i=1}^{g_v} a_i v_i - \sum_{j=1}^{g_h} b_j h_j \end{aligned} \tag{1}$$

where w_{ij} represents the symmetric interaction term between visible unit i and hidden unit j, while b_i and a_j are bias terms for hidden units and visible units, respectively.

According to energy function, the joint probability distribution of RBM model with visible and hidden units is defined.

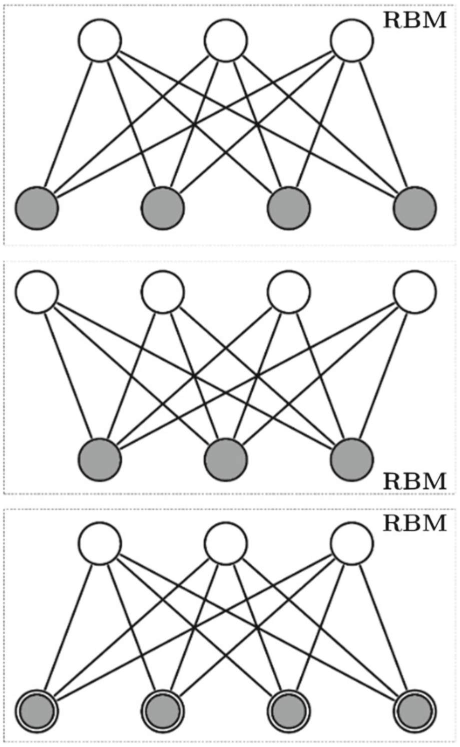


Fig. 1 Multiple layers of RBMs. Outputs of hidden layer in each RBM can be considered as input for the next RBM layer. *Shaded nodes* are visible units

$$P(\boldsymbol{v}) = \sum_{\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{h} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right) \tag{2}$$

where Z is partition function or normalization constant. Since there are no connections between neurons in the same layer, the conditional distributions are factorial and are given by:

$$P\left(h_j = 1 \middle| \boldsymbol{v}\right) = g\left(b_j + \sum_i v_i w_{ij}\right) \tag{3}$$

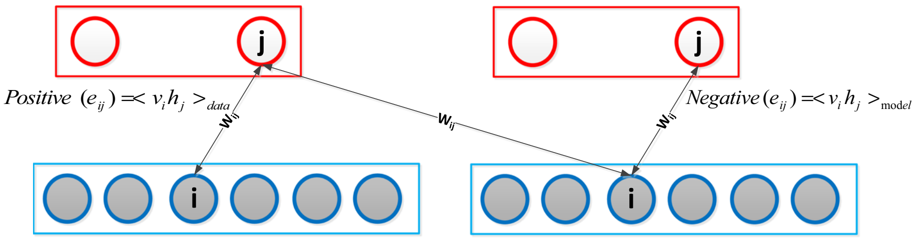
$$P\left(v_i = 1 \middle| \boldsymbol{h}\right) = g\left(a_i + \sum_j h_j w_{ij}\right) \tag{4}$$

where $g(.)$ is the logistic sigmoid function [18].

Training in RBMs maximizes probability distribution in training data with respect to the model parameters.

$$\text{maximize}_{\{w_{ij}, a_i, b_j\}} \frac{1}{m} \sum_{l=1}^{m} \log\left(P\left(\boldsymbol{v}^{(l)}\right)\right) \tag{5}$$

where the parameter m is the number of training data samples and $\boldsymbol{W}$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are RBM parameters. To maximize probability distribution, these RBM parameters need to be learnt. Because of the presence of Z, gradient calculation is not possible. Therefore, sampling methods are used in gradient calculation.



j
j
Positive $(e_{ij}) =< v_i h_j >_{data}$
w_{ij}
w_{ij}
w_{ij}
Negative $(e_{ij}) =< v_i h_j >_{model}$
i
i

Fig. 2 Visualization of 100 bases learned by the second hidden layer in DBN (*left*) and nsDBN (*right*) from the MNIST digit dataset. Here, each second layer base was visualized as a weighted linear combination of the first layer bases

$$\frac{\partial \log P(\boldsymbol{v})}{\partial w_{ij}} =< v_i h_j >_{\text{data}} - < v_i h_j >_{\text{model}} \tag{6}$$

The expectation $< v_i h_j >_{\text{data}}$ is the expectation observed in the training set (with h_j sampled given v_i according to the model), and $< v_i h_j >_{\text{model}}$ is the expectation under the distribution defined by the model [19]. Since complete computing of $< v_i h_j >_{\text{model}}$ is extremely time-consuming, so other sampling methods such as contrastive divergence (CD) [7], persistent contrastive divergence (PCD) [23] and free energy in persistent contrastive divergence (FEPCD) [11] can be used instead. Computation steps in the CD1 method are illustrated in Fig. 2.

3 Sparse RBM

Different methods are proposed to build sparse RBMs [9,14,21]. Essentially, RBMs learn non-sparse distributed representations. In all proposed methods, the learning algorithm in RBM has been changed to enforce RBM to learn sparse representation. The goal of sparsity in RBM is that most hidden units have zero values or in other words, the activation probability of hidden units be close to zero. For this purpose, a regularization term (L_{sparsity}) is defined to reduce the average activation probability on the training data. This regularization ensures that the "firing rate" of the model neurons (corresponding to the latent random variables h_j) is kept at a fairly low level, so that the activations of the model neurons are sparse [14]. Thus, given a training set $\{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(m)}\}$ comprising m examples, the optimization problem in (5) will be changed by adding the regularization term:

$$\text{maximize}_{\{w_{ij}, a_i, b_j\}} \frac{1}{m} \sum_{l=1}^{m} \log \left(P \left(\boldsymbol{v}^{(l)} \right) \right) + \lambda L_{\text{sparsity}} \tag{7}$$

where λ is a regularization constant. Thus, our objective is the sum of a log-likelihood term and a regularization term (i.e., a tradeoff between "likelihood" and "sparsity"). In principal, we can apply gradient descent to this problem; however, computing the gradient of the log-likelihood term is expensive. Fortunately, an efficient approximation to the gradient of the log likelihood like CD, PCD or FEPCD can be used. Therefore, in each iteration we can apply the approximation to the gradient of the log-likelihood update rule, followed by one step of gradient descent using the gradient of the regularization term, as summarized in Algorithm 1.

Algorithm 1 Sparse RBM learning algorithm [15]

1. *Update the parameters* ($\boldsymbol{W}$,$\boldsymbol{a}$,$\boldsymbol{b}$) *using approximation to the gradient of the log likelihood like CD, PCD or FEPCD.*
2. *Update the parameters* ($\boldsymbol{W}$,$\boldsymbol{a}$,$\boldsymbol{b}$) *using the gradient of the regularization term.*
3. *Repeat steps 1 and 2 until convergence or reach the max epoch.*

Different researchers have used different regularization terms. One of them penalizes quadratically as a deviation of the expected activation of the hidden units from a (low) fixed level p [15] (we name it quadratic sparse RBM or qsRBM).

$$L_{\text{sparsity}} = -\sum_{j=1}^{n} \left| p - \frac{1}{m} \sum_{l=1}^{m} \mathbb{E}\left[h_j^{(l)} | \boldsymbol{v}^{(l)} \right] \right|^2 \tag{8}$$

where $\mathbb{E}[.]$ is the conditional expectation on the jth hidden unit given the data and p is a constant controlling the sparseness of the n hidden units h_j. Most researchers use this method as sparsity technique. Another paper used the cross entropy between the desired (p) and actual $\left(\frac{1}{m} \sum_{l=1}^{m} \mathbb{E}\left[h_j^{(l)} | \boldsymbol{v}^{(l)} \right] \right)$ distributions of activations [21]. Due to identical equations of this method and qsRBM, we avoid expressing it for brevity.

In another paper based on rate distortion theory, the penalty factor is the activation probability of hidden units [9] (we call it rate distortion sparse RBM or rdsRBM).

$$L_{\text{sparsity}} = -\sum_{l=1}^{m} \left\| P\left(\boldsymbol{h}^{(l)} | \boldsymbol{v}^{(l)} \right) \right\|_1 \tag{9}$$

In another paper, the authors offered a theoretical approach for sparse constraints in the DBN using the mixed norm for both nonoverlapping and overlapping groups [5].

$$L_{\text{sparsity}} = -\sum_{M} \left\| P\left(\boldsymbol{h}^{(l)} | \boldsymbol{v}^{(l)} \right) \right\|_2 \tag{10}$$

where M is the number of groups formed in it. In a similar paper [17], without using groups, the logarithm of output value has been used.

$$L_{\text{sparsity}} = -\sum_{l=1}^{m} \log \left(1 + P\left(\boldsymbol{h}^{(l)} | \boldsymbol{v}^{(l)} \right)^2 \right) \tag{11}$$

Finally, among these different methods, one more cited (qsRBM) and the most recent one (rdsRBM) are selected to be compared with the proposed method. In the following, we explain a new regularization term with different properties that improves the classification results.

4 Normal sparse RBM (nsRBM)

In this paper, we proposed a new regularization term that has different behavior according to deviation of the activation of the hidden units from a (low) fixed level p . We used normal probability density function as the regularization term.

$$L_{\text{sparsity}} = \sum_{j=1}^{n} f\left(q_j, p, \sigma^2 \right) \tag{12}$$

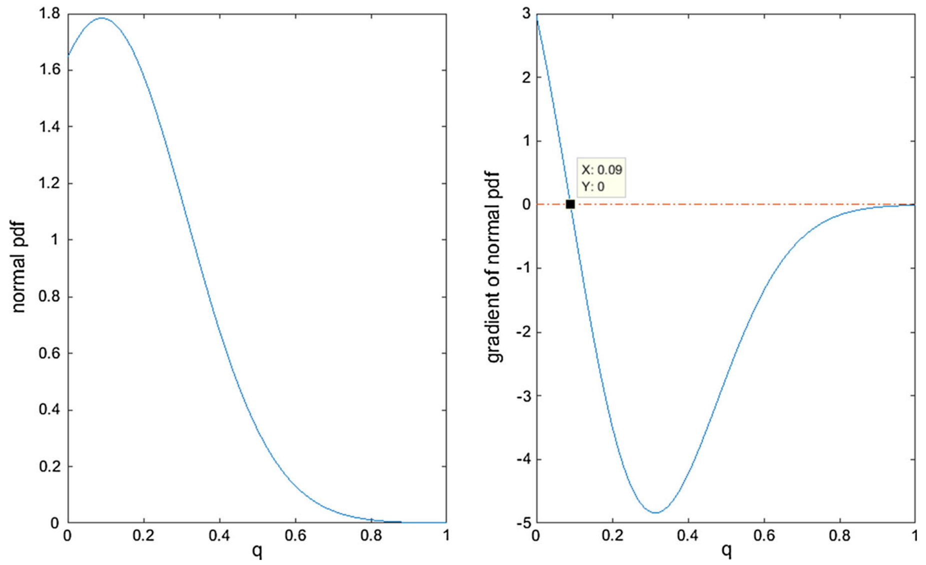


Fig. 3 *Left-hand side figure* normal pdf with $p = 0.09$ and $\sigma^2 = 0.05$ in the interval of [0, 1]. *Right-hand side figure* the gradient of the first figure. This figure shows different behaviors according to the difference between activation of hidden units and a target value (p)

where f is a normal probability density function, q_j is the average of conditional expectation on the jth hidden unit given the data, p is a constant (low) fixed level that controls the sparseness and σ^2 is variance.

$$f\left(q_j, p, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{q_j - p}{\sigma}\right)^2} \tag{13}$$

where σ is standard deviation [10].

$$q_j = \frac{1}{m}\sum_{l=1}^{m} \mathbb{E}\left[h_j^{(l)} | \boldsymbol{v}^{(l)}\right] \xrightarrow{h_j \text{ can only take values 0 or 1}}$$

$$q_j = \frac{1}{m}\sum_{l=1}^{m} P\left(h_j^{(l)} = 1 | \boldsymbol{v}^{(l)}\right) = \frac{1}{m}\sum_{l=1}^{m} q_j^{(l)} \tag{14}$$

In the above equation $\mathbb{E}\,[.]$ is the conditional expectation on the jth hidden unit given the data, m is the number of training set and $q_j^{(l)} = g(I_j)$ is equal to the activation probability of the hidden unit h_j given $\boldsymbol{v}$ and g is the logistic function. This penalty function and its derivative are depicted in Fig. 3.

According to Fig. 3, maximum updates in parameters (or maximum of absolute value of gradient) occur when activation probability in the hidden units is not near zero or one. This figure shows that when activation probability is near the mean of normal pdf, which is near zero, parameter updates are inclined to zero and the hidden unit acts well. When activation probability is near one, the unit may indicate an important factor that must be active and therefore, we penalize it only a little bit. If this hidden unit does not indicate an important factor, after some epochs, the activation probability deviates from one. In addition, zero value is not a suitable value because this means that the unit is dead and does not work properly.

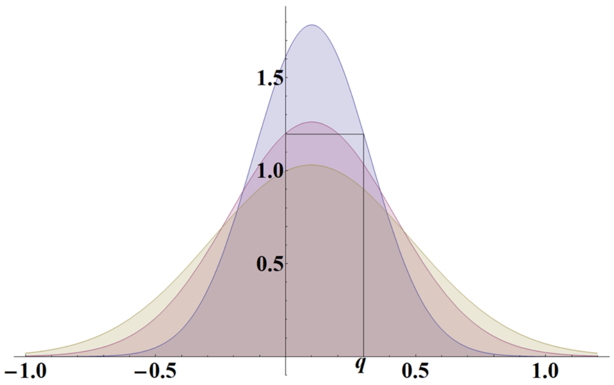


Fig. 4 Normal pdf with $p = 0.1$ and different variance values ($\sigma^2 = 0.05, \sigma^2 = 0.1, \sigma^2 = 0.15$). Penalty function can be controlled by different variance values

Finally, if activation probability in the hidden units is not near zero or one, which means that the unit cannot show a specific factor and has little information, the gradient will update parameters exponentially to have activation probability near zero value.

In addition, our proposed regularization term has a parameter that can control the force degree of sparseness. According to Fig. 4 by different variance values, we can control the penalty value or degree of sparseness.

According to Algorithm 1, the gradient of regularization term must be used in step 2. The gradient of our regularization term can be computed as follows:

$$\frac{\partial}{\partial w_{ij}} L_{\text{sparsity}} = \frac{\partial \left(f\left(q_j, p, \sigma^2\right)\right)}{\partial w_{ij}} = \frac{\partial \left(f\left(q_j, p, \sigma^2\right)\right)}{\partial q_j} \times \frac{\partial \left(q_j\right)}{\partial I} \times \frac{\partial \left(I\right)}{\partial w_{ij}} \tag{15}$$

where $I_j = b_j + \sum_i v_i w_{ij}$ is equal to the sum of inputs to hidden unit j.

$$\frac{\partial \left(f\left(q_j, p, \sigma^2\right)\right)}{\partial q_j^{(l)}} = \frac{\left(p - q_j\right)}{\sigma^2} f\left(q_j, p, \sigma^2\right) \tag{16}$$

$$\frac{\partial \left(q_j\right)}{\partial I} \times \frac{\partial \left(I\right)}{\partial w_{ij}} = \frac{1}{m} \sum_{l=1}^{m} q_j^{(l)} \left(1 - q_j^{(l)}\right) v_i^{(l)} \tag{17}$$

where $q_j^{(l)} = g(I_j)$ is equal to activation probability of hidden unit h_j given $\boldsymbol{v}$ and g as the logistic function. Therefore, the RBM parameters are updated with the regularization term based on the following equations:

$$\frac{\partial}{\partial w_{ij}} L_{\text{sparsity}} \propto \frac{1}{m} \left(p - \frac{1}{m} \sum_{l=1}^{m} q_j^{(l)}\right) f\left(q_j, p, \sigma^2\right) \sum_{l=1}^{m} q_j^{(l)} \left(1 - q_j^{(l)}\right) v_i^{(l)} \tag{18}$$

Finally, for hidden bias we have the following equation:

$$\frac{\partial}{\partial b_j} L_{\text{sparsity}} \propto \frac{1}{m} \left(p - \frac{1}{m} \sum_{l=1}^{m} q_j^{(l)}\right) f\left(q_j, p, \sigma^2\right) \sum_{l=1}^{m} q_j^{(l)} \left(1 - q_j^{(l)}\right) \tag{19}$$

Finally, by comparing the proposed method with other methods, it can be concluded that this method has some features that is rarely seen in other methods. One of them is that other methods have an invariant form in their regularization term but the regularization term in nsDBN can be changed and can control the force degree of sparseness. This advantage enables better controlling of the force degree of sparseness. The other benefit of our method is that the new regularization term has different behaviors according to deviations of the activation of the hidden units from a (low) fixed value. According to these qualifications, it will be depicted in the following experiments that our proposed method outperforms other methods.

5 Results

The method proposed in this paper was evaluated on different datasets with different applications and input type. These datasets are in the tasks of digit recognition on the MNIST dataset with discrete feature values between 0 and 255, spoken letter recognition on the ISOLET dataset with real feature values and a document topic classification on the 20 Newsgroups dataset with binary feature values. Also, we used the DeeBNet MATLAB toolbox [12] (the toolbox implemented by the authors) in the experiments. In addition, we used the new FEPCD method [17] to approximate the gradient of the log likelihood (see Sect. 2). Our nsRBM method has been implemented and is available online.[1]

5.1 MNIST dataset

MNIST[2] dataset includes images of handwritten digits [13] (10 classes of digits 0–9). Each digit was carefully located in the center of each 28*28 image. The image pixels have discrete values between 0 and 255, most of which have the values at the edge of this interval [20]. The dataset was divided to train and test parts including 60,000 and 10,000 images, respectively. In our experiments, these discrete values were mapped to the interval [0–1] using the min–max normalization method as following:

$$x_i' = \frac{x_i - \min(X)}{\max(X) - \min(X)} \tag{20}$$

where x_i is a sample attribute value, $\min(X)$ is the minimum value of entire dataset (in the MNIST dataset, this value is 0), $\max(X)$ is the maximum value of entire dataset (in the MNIST dataset, this value is 255), and x_i' is the normalized value.

As the first result, we used an RBM and a normal sparse RBM with 500 hidden units. Figure 5 shows a histogram of activation probability of hidden units for one image. According to this figure, normal sparse RBM can reduce activation of hidden units significantly.

In another test with the same RBM and nsRBM, the bases in nsRBM are more informative. Figure 6 shows that the resulted bases learnt by nsRBM have a clear structure and in the nsRBM method, the greatest proportion of interesting feature detectors was learnt.

After learning an RBM with 500 hidden units, we learnt a new RBM and nsRBM with 100 hidden units on the activation probability of hidden units produced by RBM and nsRBM, respectively. These stacked RBMs and nsRBMs composed a DBN and an nsDBN. The bases generated by the second hidden layer are visualized in Fig. 7.

[1] Available online at "http://ceit.aut.ac.ir/~keyvanrad/DeeBNetToolbox.html".

[2] Available online at "http://yann.lecun.com/exdb/mnist/".

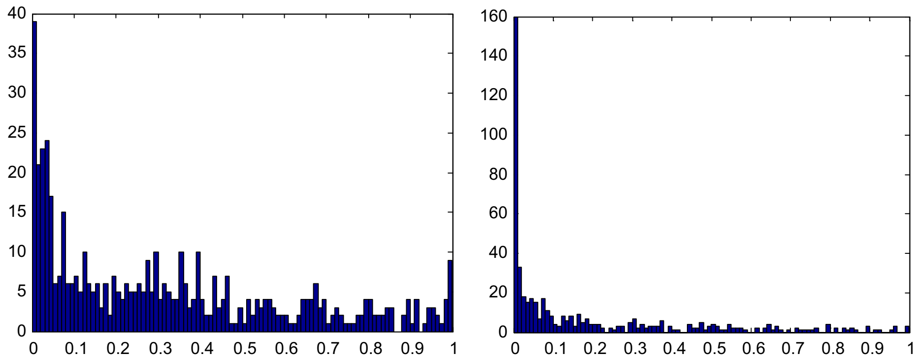


Fig. 5 Activation probability of hidden units for one image obtained by RBM (*left*) and normal sparse RBM (*right*). According to this figure, normal sparse RBM can significantly reduce activation of hidden units

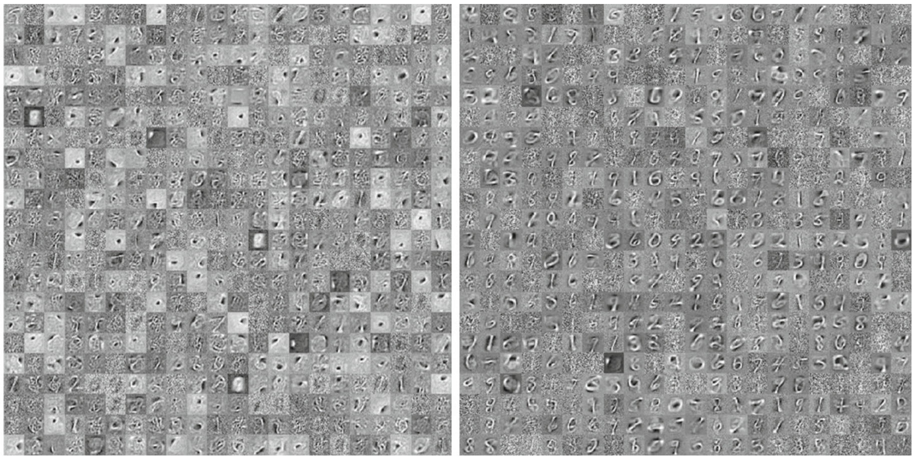


Fig. 6 Visualization of 484 bases learned by RBM (*left*) and nsRBM (*right*) from the MNIST digit dataset. Here bases are weights corresponding to each pixel in each hidden unit. Bases in nsRBM are more informative and have a clear structure compared to classic RBM

According to Fig. 7, the bases of the second hidden layer learnt by nsDBN are more abstract than those obtained by DBN. It is obvious that they learned the digits distinctly and can capture higher order correlations among the input pixel intensities.

For a better comparison and to demonstrate the superiority of this method in learning features, several experiments have been conducted. In these experiments, our proposed method is compared with two other sparsity methods (qsRBM and rdsRBM), classic RBM, principal component analysis and raw features. Also, all models have been trained on 20,000 training samples of MNIST dataset (similar to [9]) to extract 500 features from visible data (except in raw features where all input features are used).

Now with these learnt models, features of 10, 20, 50, 100, 500 and 1000 images per class are used in training a linear classifier. The reason of using a simple linear classifier is to achieve a better comparison between strength of learnt features. Also for better classifier evaluation, image selection in each class and linear classifier training has been run twenty times separately and averages on results have been reported.

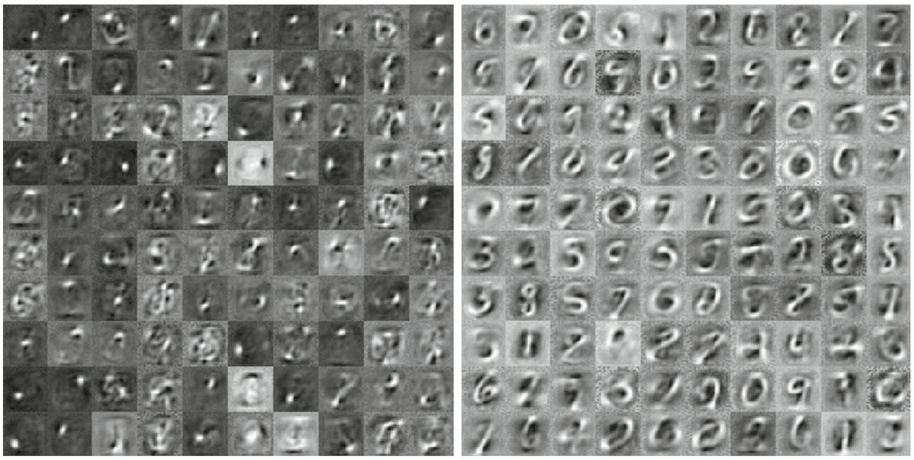


Fig. 7 Visualization of 100 bases learned by the second hidden layer in DBN (*left*) and nsDBN (*right*) from the MNIST digit dataset. Here, each second layer base is visualized as a weighted linear combination of the first layer bases

Table 1 Digit recognition error rate on MNIST dataset on training (with 10, 20, 50, 100, 500 and 1000 samples per class) and test sets, and by a linear classifier that has been trained on active probability of hidden units produced by rate distortion sparse RBM (rdsRBM) using four best parameters

Parameters (rdsRBM)			Number of samples					
λ	p	σ^2	10	20	50	100	500	1000
0.02	–	–	**16.240**	**11.482**	**7.996**	6.520	4.401	3.849
0.01	–	–	16.674	11.732	8.151	6.612	4.374	3.685
0.005	–	–	16.446	11.798	8.013	**6.379**	**4.121**	**3.562**
0.001	–	–	16.817	11.972	8.302	6.564	4.274	3.601

In order to facilitate the comparison, the best results were highlighted in bold

Table 1 displays error rate on the MNIST test set produced by a linear classifier trained on active probability of hidden units produced by rdsRBM with the four best parameters. Also, Tables 2 and 3 display error rate on qsRBM and nsRBM, respectively, with four best parameter combinations. In our experiments, we did a simple grid search for about three parameters using different grid points per parameter. For example, in MNIST dataset, all combination of five different values for cost value (in the interval [0.1, 3]), six different values for target value (in the interval [0.01, 0.1]) and three different values for variance (in the interval [0.05, 0.15]) were tested on nsRBM method and finally, four best parameter combinations were selected.

In order to facilitate the comparison, the best results were highlighted in bold. According to these results, the nsRBM has the best results under each training sample size.

In Table 4, the recognition ability based on raw data (784 features), transformed codes produced by PCA (500 features), RBM, qsRBM, rdsRBM and nsRBM (all with 500 hidden units) are compared. In this table, the results obtained by the best parameter combinations for each of the qsRBM, rdsRBM and nsRBM are presented. According to Table 4, it can be seen that nsRBM always achieves the best recognition accuracy on the MNIST test set for all different numbers of samples.

Table 2 Digit recognition error rate on MNIST dataset training (with 10, 20, 50, 100, 500 and 1000 samples per class) and test sets, and by a linear classifier that has been trained on active probability of hidden units produced by quadratic sparse RBM (qsRBM) using four best parameters combination

Parameters (qsRBM)			Number of samples					
λ	p	σ^2	10	20	50	100	500	1000
2	0.1	–	15.864	10.857	7.252	5.891	3.799	3.221
1	0.07	–	15.713	10.855	7.290	**5.789**	3.783	**3.165**
1	0.03	–	**15.040**	**10.376**	**7.061**	**5.789**	3.956	3.428
0.5	0.07	–	16.128	10.885	7.452	5.835	**3.780**	3.171

In order to facilitate the comparison, the best results were highlighted in bold

Table 3 Digit recognition error rate on MNIST dataset training (with 10, 20, 50, 100, 500 and 1000 samples per class) and test sets, and by a linear classifier that has been trained on active probability of hidden units produced by normal sparse RBM nsRBM using four best parameters combination

Parameters (nsRBM)			Number of samples					
λ	p	σ^2	10	20	50	100	500	1000
3	0.09	0.05	15.031	10.262	**6.810**	**5.483**	**3.633**	**3.081**
3	0.07	0.05	14.758	**9.894**	7.054	5.568	3.698	3.128
2	0.07	0.1	**14.563**	10.518	6.956	5.545	3.747	3.199
0.5	0.05	0.1	15.733	10.418	7.083	5.573	3.741	3.253

In order to facilitate the comparison, the best results were highlighted in bold

Table 4 Digit recognition error rate on MNIST dataset on training (with 10, 20, 50, 100, 500 and 1000 samples per class) and test sets, and by a linear classifier that has been trained on raw data, transformed codes produced by PCA and active probability of hidden units produced by RBM, qsRBM, rdsRBM and nsRBM

Methods (features)	Number of samples					
	10	20	50	100	500	1000
RAW	23.100	17.416	12.806	10.767	7.646	6.700
PCA	22.546	17.199	12.990	10.706	7.596	6.719
RBM	17.734	12.291	8.567	6.825	4.445	3.734
qsRBM	15.040	10.376	7.061	5.789	3.956	3.428
rdsRBM	16.446	11.798	8.013	6.379	4.121	3.562
nsRBM	**15.031**	**10.262**	**6.810**	**5.483**	**3.633**	**3.081**

In order to facilitate the comparison, the best results were highlighted in bold. It can be seen that nsRBM always achieves the best recognition accuracy on the MNIST test set for all different number of samples

A notable point in these results is the superiority of features obtained from all sparse methods compared to other non-sparse features. However, this result was predictable because as discussed before, according to learning theory, to obtain a good generalization, it is enough that the total number of bits needed to encode the whole training set be small, compared to the size of the training set [2]. This property can be obtained by sparsity. In addition, these sparse features are closer to brain function, because it seems that our brains use a representation that is sparse and only 1–4% of the neurons are active at a given time [2].

Table 5 Digit recognition error rate on MNIST training (with 10, 20, 50, 100, 500 and 1000 samples per class) and test sets, and by a linear classifier that has been trained on active probability of hidden units produced by the first and second layers in DBN, qsDBN, rdsDBN and nsDBN

Method	Number of samples					
	10	20	50	100	500	1000
DBN	15.947	11.321	7.958	6.440	4.102	3.478
qsDBN	13.929	9.483	6.728	5.577	3.777	3.233
rdsDBN	15.123	10.738	7.372	5.938	3.840	3.265
nsDBN	**13.368**	**9.291**	**6.508**	**5.263**	**3.418**	**2.899**

The best results are highlighted in bold. nsDBN always achieves the best recognition accuracy on the training and test sets for different number of samples

Another notable point in these results is that our method outperforms all other methods. This superiority is due to the regularization term that has different behaviors according to the difference between activation value of hidden units and a (low) fixed value. In addition, the regularization term has a variance parameter that can control the force degree of sparseness.

Since a DBN can learn higher levels of representation in its layers [2], in the next experiments, a layer with 100 hidden neurons was added to last models (RBM, qsRBM, rdsRBM and nsRBM). After learning new models with their learning methods, a feature vector of learnt features in first and second layer was used for training linear classifier (500 and 100 features in first and second layers, respectively). Table 5 displays the error rate on the test set. According to Table 5, we can see again that nsDBN always achieves the best recognition accuracy on the training and test sets when using a different number of samples.

5.2 ISOLET dataset

In the ISOLET[3] dataset, 150 speakers utter the name of each of 26 types of letters of the alphabet twice. Hence, there are 52 training examples from each speaker. The speakers are grouped into sets of 30 speakers each and are referred to as isolet1, isolet2, isolet3, isolet4 and isolet5. The train data appear in the isolet1+2+3+4 data file in sequential order, i.e., first the speakers from isolet1, then isolet2, and so on. The test set, isolet5, is a separate file. Due to three missing examples, there are 7797 examples in total referred to as isolet1-isolet5 (6238 training examples and 1559 test examples). The feature vector has 617 features including spectral coefficients, contour features, sonorant features, pre-sonorant features and post-sonorant features [3].

Since features have real values, the Gaussian visible units are used [6]. As the first result in the ISOLET dataset, we used an RBM and a normal sparse RBM with 500 hidden units. Figure 8 shows the histogram of activation probability of hidden units for all test data. According to this figure, normal sparse RBM can reduce activation of hidden units significantly.

Similar to the MNIST test, our proposed method is compared with two other sparsity methods (qsRBM and rdsRBM), classic RBM, principal component analysis and raw features. Also, all models have been trained on 3640 feature vectors (i.e., 140 feature vectors per class) of ISOLET dataset to extract 500 features from visible data (except in raw features where all input features are used).

[3] Available online at "https://archive.ics.uci.edu/ml/datasets/ISOLET".

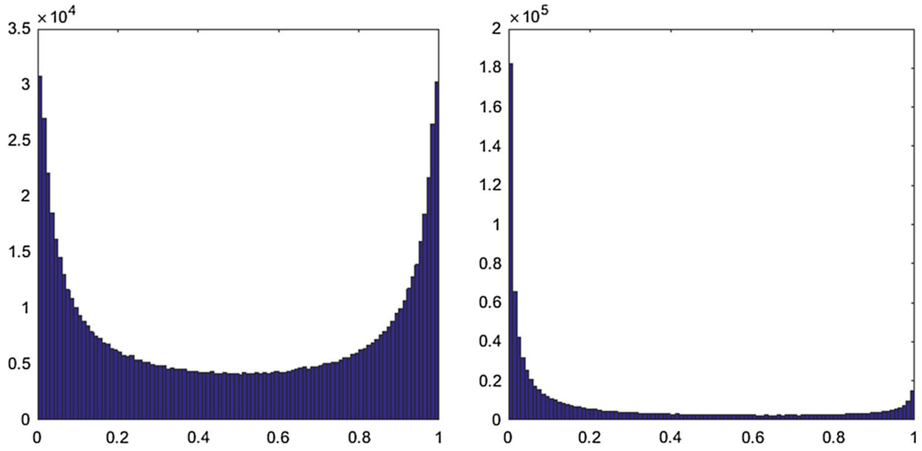


Fig. 8 Histogram of activation probability of hidden units for all test data in ISOLET dataset by RBM (*left*) and normal sparse RBM (*right*). Normal sparse RBM reduces significantly the activation of hidden units

Table 6 Spoken letter recognition error rate on ISOLET dataset on training (with 10, 20, 50 and 100 samples per class) and test sets, and by a linear classifier that has been trained on raw data, transformed codes produced by PCA and active probability of hidden units produced by RBM, qsRBM, rdsRBM and nsRBM

Method	Number of samples			
	10	20	50	100
RAW	13.836	8.996	6.084	5.413
PCA	14.034	9.573	6.565	5.455
RBM	14.082	9.531	6.247	5.282
qsRBM	13.162	8.929	5.956	**4.278**
rdsRBM	13.807	9.051	6.071	4.676
nsRBM	**12.742**	**8.450**	**5.930**	4.740

The best results are highlighted in bold

Now with these learnt models, features of 10, 20, 50 and 100 samples per class are used in training a linear classifier. For each combination of training data size and algorithm, we trained 20 classifiers with randomly chosen training sets for the linear classifier and then used the average classification error to evaluate the performance of the corresponding algorithm. Similar to the MNIST test, several values of hyper-parameters were chosen to conduct experiments. For a summary of results, only the best model was reported.

In Table 6, the recognition ability based on raw data (617 features), transformed codes produced by PCA (500 features), RBM, qsRBM, rdsRBM and nsRBM (all with 500 hidden units) are compared. In this table, the results obtained by the best parameter combinations for each of qsRBM, rdsRBM and nsRBM are presented. In order to facilitate the comparison, the best results were highlighted in bold. According to Table 6, it can be seen that nsRBM in most cases achieves the best recognition accuracy on the ISOLET test set.

Similar to experiments done on MNIST, in order to compare the recognition ability of representations learnt by DBN, qsDBN, rdsDBN and nsDBN, a layer with 100 hidden neurons was trained. After learning new models with their learning methods, a feature vector of learnt features in first and second layer was used for training linear classifier (500 and 100 features in

Table 7 Spoken letter recognition error rate on ISOLET training (with 10, 20, 50 and 100 samples per class) and test sets, produced by a linear classifier that has been trained on active probability of hidden units produced by the first and second layer in DBN, qsDBN, rdsDBN and nsDBN

Method	Results with different number of samples			
	10	20	50	100
DBN	13.239	8.874	5.991	5.109
qsDBN	12.758	8.448	5.892	4.262
rdsDBN	13.435	8.958	5.859	4.596
nsDBN	**12.473**	**8.287**	**5.795**	**4.057**

The best results are highlighted in bold

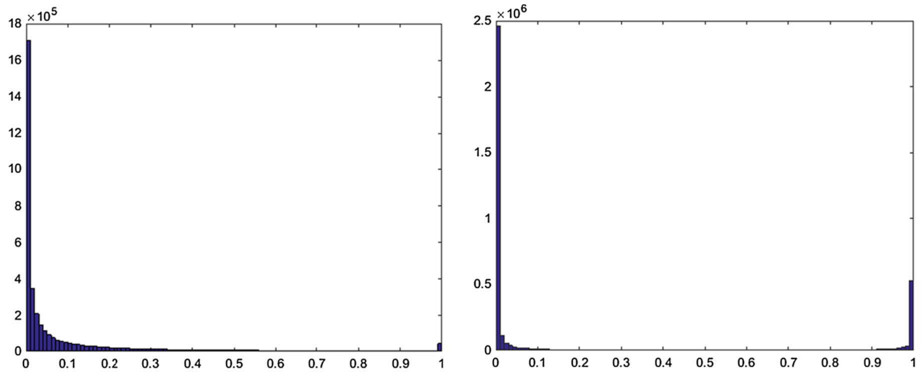


Fig. 9 Histogram of activation probability of hidden units for all test data in 20 Newsgroups dataset by RBM (*left*) and normal sparse RBM (*right*). Normal sparse RBM reduces activation of hidden units

first and second layer, respectively). Table 7 displays the error rate on the test set. According to Table 7, it can again be seen that nsDBN always achieves the best recognition accuracy on the training and test sets when using a different number of samples.

5.3 20 Newsgroups dataset

The 20 Newsgroups[4] dataset is organized into 20 different news groups, each corresponding to a different topic. The 20 newsgroups dataset has become a popular dataset for experiments in text applications of machine learning techniques, such as text classification and text clustering.

For a text classification experiment, we used a version of the 20 Newsgroups[5] with 18774 documents in which the training and test sets (60 and 40%, respectively) containing documents collected at different times. We used the 5000 most frequent words for binary input features.

Since the features have binary values, the binary visible units are used. As the first result in the 20 Newsgroups dataset, we used an RBM and a normal sparse RBM with 500 hidden units. Figure 9 shows the histogram of activation probability of hidden units for all test data. According to this figure, normal sparse RBM can reduce activation of hidden units.

4 Available online at "http://qwone.com/~jason/20Newsgroups".

5 Available online at "http://qwone.com/~jason/20Newsgroups/20news-bydate-matlab.tgz".

Table 8 Document topic classification error rate on 20 Newsgroups dataset on training (with 10, 20, 50, 100, 200 and 350 samples per class) and test sets, and by a linear classifier that has been trained on raw data, transformed codes produced by PCA and active probability of hidden units produced by RBM, qsRBM, rdsRBM and nsRBM

Method	Number of training samples					
	10	20	50	100	200	350
RAW	67.847	59.093	47.819	40.474	35.091	31.425
PCA	68.010	60.360	51.236	44.596	39.405	36.025
RBM	47.644	43.254	38.775	35.889	33.872	32.620
qsRBM	47.403	42.091	37.747	35.492	33.058	31.295
rdsRBM	47.242	42.617	38.787	36.362	34.466	33.310
nsRBM	**46.490**	**41.955**	**37.724**	**35.263**	**32.959**	**30.990**

The best results are highlighted in bold

Table 9 Document topic classification error rate on 20 Newsgroups dataset on training (with 10, 20, 50, 100, 200 and 350 samples per class) and test sets, and by a linear classifier that has been trained on active probability of hidden units produced by the first and second layer in DBN, qsDBN, rdsDBN and nsDBN

Method	Number of training samples					
	10	20	50	100	200	350
DBN	46.460	42.644	38.939	36.312	34.315	32.853
qsDBN	46.424	41.705	**37.345**	**34.933**	32.695	31.083
rdsDBN	45.9423	42.171	38.468	36.223	34.133	32.934
nsDBN	**45.433**	**40.977**	37.382	34.995	**32.620**	**30.953**

The best results are highlighted in bold

Similar to the MNIST test, our proposed method is compared with two other sparsity methods (qsRBM and rdsRBM), classic RBM, principal component analysis and raw features. Also, all models have been trained on 7500 feature vectors (i.e., 375 feature vectors per class) of 20 Newsgroups dataset to extract 500 features from visible data (except in raw features that all input features are used).

Now with these learnt models, features of 10, 20, 50,100,200 and 350 feature vectors per class are used in training a linear classifier. For each combination of training data size and algorithm, we trained 20 classifiers with randomly chosen training sets for the linear classifier and then used the average classification error to evaluate the performance of the corresponding algorithm. Similar to the MNIST test, several values of hyper-parameters were chosen to conduct experiments. For a summary of results, only the best model was reported.

In Table 8, the recognition ability based on raw data (5000 features), transformed codes produced by PCA (500 features), RBM, qsRBM, rdsRBM and nsRBM (all with 500 hidden units) are compared. In this table, the results obtained by the best parameter combinations for each of qsRBM, rdsRBM and nsRBM are presented. In order to facilitate the comparison, the best results were highlighted in bold. According to Table 8, it can be seen that nsRBM achieves the best recognition accuracy on the 20 Newsgroups test set in all cases. In particular, the performance improvement in small training sets is more significant.

Similar to the MNIST test, in order to compare the recognition ability of representations learnt by DBN, qsDBN, rdsDBN and nsDBN, we trained a layer with 100 hidden neurons. After learning new models with their learning methods, a feature vector of learnt features in first and second layer has been used for training linear classifier (500 and 100 features in first and second layer, respectively). Table 9 displays the error rate on the test set. According to Table 9, we can see again that in most cases, nsDBN achieves the best recognition accuracy on the test sets when using a different number of training samples.

6 Conclusion

In this paper, we discussed a new sparse RBM method. Different methods are proposed to build sparse RBMs. In this paper, we proposed a new method that has different behaviors according to a deviation of the activation of the hidden units from a (low) fixed value and in addition, our proposed regularization term has a variance parameter that can control the force degree of sparseness. According to the results, our new method achieves the best recognition accuracy on the test sets when using a different number of training samples even when we have only a few labeled samples per class. For better comparison, we used three different datasets in different applications including MNIST for image recognition, ISOLET for speech recognition and 20 Newsgroups for text categorization.

In addition, we observed that our proposed method (normal sparse RBM or nsRBM) can reduce activation of hidden units significantly. According to the results, bases that have been learnt by nsRBM have a clear structure and in the nsRBM method a much larger proportion of interesting feature detectors were learnt. In addition, the recognition ability based on raw data, PCA, RBM, qsRBM, rdsRBM and nsRBM are compared and nsRBM achieves the best recognition accuracy on the test sets when using a different number of samples.

For future work, we would like to focus on controlling the force degree of sparseness in the proposed method. This can be done by gradually decreasing the variance. This means that in the first epochs we set a large variance that has a small force on sparsity and by increasing the epochs we increase the sparsity force by decreasing the variance (see Fig. 4). Another improving idea is to use different variance in different feature directions (or hidden units). In this new idea, we set the variance according to distribution of activation probability in hidden units.

References

1. Bai L, Li K, Pei J, Jiang S (2015) Main objects interaction activity recognition in real images. Neural Comput Appl 27:335–348. doi:10.1007/s00521-015-1846-7
2. Bengio Y (2009) Learning deep architectures for AI. Found Trends Mach Learn 2:1–127
3. Fanty M, Cole RA (1990) Spoken letter recognition. In: Lippmann RP, Moody JE, Touretzky DS (eds) Proceedings of the 3rd international conference on neural information processing systems (NIPS'90). Morgan Kaufmann Publishers Inc., San Francisco, CA, pp 220–226
4. Fischer A, Igel C (2014) Training restricted Boltzmann machines: an introduction. Pattern Recognit 47:25–39. doi:10.1016/j.patcog.2013.05.025
5. Halkias X, Paris S, Glotin H (2013) Sparse penalty in deep belief networks: using the mixed norm constraint. arXiv:1301.3533 [cs, stat]
6. Hinton G (2010) A practical guide to training restricted Boltzmann machines. Machine Learning Group, University of Toronto
7. Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Comput 18:1527–1554
8. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313:504–507
9. Ji N-N, Zhang J-S, Zhang C-X (2014) A sparse-response deep belief network based on rate distortion theory. Pattern Recognit 47:3179–3191
10. Keyvanrad MA, Homayounpour MM (2015a) Normal sparse deep belief network. In: 2015 international joint conference on neural networks (IJCNN), pp 1–7

11. Keyvanrad MA, Homayounpour MM (2015b) Deep belief network training improvement using elite samples minimizing free energy. Int J Patt Pattern Artif Intell. doi:10.1142/S0218001415510064
12. Keyvanrad MA, Homayounpour MM (2014) A brief survey on deep belief networks and introducing a new object oriented toolbox (DeeBNet). arXiv:1408.3264 [cs]
13. LeCun Y, Cortes C, Burges CJ (1998) The MNIST database of handwritten digits. The MNIST data set is available at http://yann.lecun.com/exdb/mnist/index.html
14. Lee H (2010) Unsupervised feature learning via sparse hierarchical representations. Ph.D. Thesis, Stanford University
15. Lee H, Ekanadham C, Ng A (2008) Sparse deep belief net model for visual area V2. Adv Neural Inf Process Syst 20:873–880
16. Liu Y, Zhou S, Chen Q (2011) Discriminative deep belief networks for visual data classification. Pattern Recognit 44:2287–2296
17. Marc'Aurelio Ranzato Y, Boureau L, LeCun Y (2007) Sparse feature learning for deep belief networks. Adv Neural Inf Process Syst 20:1185–1192
18. Mohamed A, Dahl G, Hinton G (2009) Deep belief networks for phone recognition. In: NIPS workshop on deep learning for speech recognition and related applications. Canada, pp 1–9
19. Murphy KP (2012) Machine learning: a probabilistic perspective. The MIT Press, Cambridge
20. Nair V (2010) Visual object recognition using generative models of images. Ph.D Thesis, University of Toronto
21. Nair V, Hinton G (2009) 3D object recognition with deep belief nets. Adv Neural Inf Process Syst 22:1339–1347
22. Olshausen BA, David JF (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381:607–609
23. Tieleman T (2008) Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th international conference on Machine learning. ACM, New York, NY, USA, pp 1064–1071

Mohammad Ali Keyvanrad received the B.S. degree in software engineering from the Amirkabir University of Technology, Tehran, Iran, in 2007. He then received his MSc and Ph.D. degrees in Artificial Intelligence from the Amirkabir University of Technology, Tehran, Iran, in 2010 and 2016, respectively. His research interests include pattern recognition, machine learning and signal processing, especially deep learning, feature learning and audio indexing.

Mohammad Mehdi Homayounpour was born in 1960 in Iran. He received his BSc degree in Electronics from Amirkabir University of Technology in 1986, MSc in Telecommunications from Khaje Nasireddin Toosi in 1989, and Ph.D. in Electrical Engineering from Paris-11 University, Paris, France. He has been a faculty member of Computer Engineering and IT Department at Amirkabir University of Technology (Tehran Polytechnics), Tehran, Iran, since 1995. His research interests include signal processing, speech recognition and enhancement, speaker recognition, text to speech, audio indexing, statistical methods for signal analysis and modeling, natural language processing, hardware design and multimedia.