Modeling Hebb Learning Rule for Unsupervised Learning

Jia Liu[1], **Maoguo Gong**[1*], **Qiguang Miao**[2]
[1]Key Laboratory of Intelligent Perception and Image Understanding, Xidian University, Xi'an, China
[2]School of Computer Science and Technology, Xidian University, Xi'an, China
omegaliuj@gmail.com, gong@ieee.org, qgmiao@mail.xidian.edu.cn

Abstract

This paper presents to model the Hebb learning rule and proposes a neuron learning machine (NLM). Hebb learning rule describes the plasticity of the connection between presynaptic and postsynaptic neurons and it is unsupervised itself. It formulates the updating gradient of the connecting weight in artificial neural networks. In this paper, we construct an objective function via modeling the Hebb rule. We make a hypothesis to simplify the model and introduce a correlation based constraint according to the hypothesis and stability of solutions. By analysis from the perspectives of maintaining abstract information and increasing the energy based probability of observed data, we find that this biologically inspired model has the capability of learning useful features. NLM can also be stacked to learn hierarchical features and reformulated into convolutional version to extract features from 2-dimensional data. Experiments on single-layer and deep networks demonstrate the effectiveness of NLM in unsupervised feature learning.

1 Introduction

A promising but challenging research in artificial intelligence has been mimicking the process the human brain represents information for decades [Arel *et al.*, 2010]. Neural science findings [Kruger *et al.*, 2013] have provided new ideas for designing information representation systems. This has motivated the development of hierarchical learning architectures and feature learning models.

Unsupervised feature learning aims to learn good representations from unlabeled input data. One of the interests in unsupervised learning is the stacked single-layer learning models for learning hierarchical representations [Hinton *et al.*, 2006; Hinton and Salakhutdinov, 2006; Bengio *et al.*, 2006; 2013]. Unsupervised learning had a catalytic effect in reviving interest in deep neural networks [LeCun *et al.*, 2015]. Although it has little effect in many purely supervised applications, it still works when large amount of labeled samples are unavailable. Various single-layer feature learning models have been used and proposed since the layer-wise training by restricted Boltzmann machine (RBM) [Hinton *et al.*, 2006] and auto-encoder (AE) [Bengio *et al.*, 2006], including principal component analysis (PCA) [Valpola, 2014], sparse coding [He *et al.*, 2013], denoising AE [Vincent *et al.*, 2010], and sparse versions of AE and RBM [Lee *et al.*, 2007; Ji *et al.*, 2014; Ranzato *et al.*, 2007; Gong *et al.*, 2015].

Good representations eliminate irrelevant variabilities of the input data, while preserving the information that is useful for the ultimate task [Ranzato *et al.*, 2007]. For this purpose, an intuitive way is to construct the encoder-decoder model. AE, RBM, PCA and most unsupervised single-layer feature learning models follow this paradigm. Based on the encoder-decoder architecture, more useful features can be learned by adding constrains to extract more abstract concepts, including sparse constraints and sensitivity constraint [Rifai *et al.*, 2011]. Another type of learning models only have a decoder, sparse coding for example. The above models can be taken as task based models which model the task and find the solution that can well handle the task. However, some tasks are amphibolous to define and model. The feature learning task for instance, there are many ways to define the abstraction, e.g., sparse representation, noise corruption [Vincent *et al.*, 2010], and contractive constraint [Rifai *et al.*, 2011].

In this paper, we propose a learning rule based model which attempts to model the learning process of neurons with less considerations of the task. The model consists of only the encoder. The learning behaviours of neurons have been researched for a long time for revealing the mechanism of human cognition. One of the most famous theory is the Hebb learning rule [Hebb, 1949] proposed by Donald Olding Hebb. By applying the Hebb rule in the study of artificial neural networks, we can obtain powerful models of neural computation that might be close to the function of structures found in neural systems of many diverse species [Kuriscak *et al.*, 2015]. Recent works on Hebb learning are the series on "biologically plausible backprop" [Scellier and Bengio, 2016] by Bengio, et al. They linked Hebb and other biologically formulated rules to back-propagation algorithm and used them to train the neural network. Hebb rule gives the updating gradient of the connecting weights and it is unsupervised itself. We establish the model with the single layer network as many unsupervised learning models and attempt to mod-

*Corresponding author

el the updating gradient as the objective function. However, the architecture of artificial neural network is highly simplified from the neural network in brain. Therefore, when the learning rule is implemented, several properties have to be considered [Kuriscak *et al.*, 2015] in order to adapt to and reinforce the simplified network. We call the model neuron learning machine (NLM) and find that NLM has many similar properties to other unsupervised feature learning models. From the analysis and experiments, NLM is capable of learning useful features and outperform the compared task based models. For learning features from 2-dimensional data, e.g., images and videos, we also extend the model into a convolutional version.

This paper is organized in five sections. In the next section, the proposed NLM is formulated. In Section 3, we explain how the NLM works and learns useful features. The experimental studies are provided in Section 4. In Section 5, we conclude this paper.

2 The Model

The structure of NLM follows the common structure of single-layer neural network, e.g., the encoder of the AE and RBM. The values of hidden units are computed by:

$$h_i = s(W_i v + b_i) \tag{1}$$

where h is the hidden vector and v is the visible vector. W and b denote the connecting weight matrix and bias vector respectively. s is the activation function and here we use the sigmoid function $s(x) = 1/(1 + \exp(-x))$.

Hebb learning rule can be summarised by the most cited sentence in [Hebb, 1949]: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." In the training of an artificial neural network, Hebb rule describes the updating gradient of each connecting weight $W_{ij} = W_{ij} + \Delta W_{ij}$. This can be simplified by the product of the values of the units at the two ends of the connection:

$$\Delta W_{ij} = \alpha h_i v_j \tag{2}$$

where α is the learning rate. Hebb rule itself is an unsupervised learning rule which formulates the learning process of neurons. Therefore, we attempt to model the network based on this learning rule in order to learn in the neuronal way.

With the gradient, a model can be obtained by integrating over ΔW. However, since W is a matrix and h actually is a function of W and v, i.e., $h = f(W, v)$, it is of great difficulty to derive the integral. As discussed above, the artificial neural networks are highly simplified models inspired from real neural networks. Therefore when the Hebb learning rule is applied to the simplified network, many considerations have to be considered, i.e., locality, cooperativity, weight boundedness, competition, long term stability, and weight decrease and increase [Kuriscak *et al.*, 2015]. With these considerations and constraints, we can simplify the integration process. In real neural networks, there are many factors influencing the

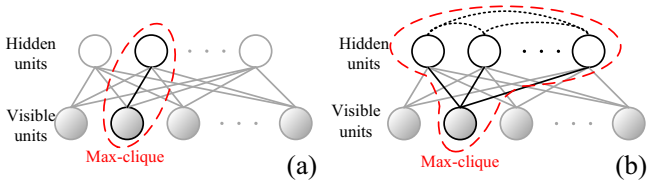

Figure 1: Undirected graphical models of network with and without competition. (a) Network without competition (b) Network with competition

status of hidden units, including status of visible units, information from other nucleus, competition between hidden neurons and adjacent connections and limitations of the neuron itself. Then, a connection of the simplified network has less influence on h. Therefore, for convenience, we assume that the connecting weight has negligible influence on the connected hidden unit. Under this omitting assumption, h can be taken as a set of constants wrt W and the integral is apparent, i.e., $\max \sum_{ij} \int_{W_{ij}} h_i v_j = \sum_{ij} W_{ij} h_i v_j = v^T W h$. Although this term is derived under the hypothesis, optimizing this term will increase the correlation between W_{ij} and $v_j h_i$ which is consistent with the Hebb rule. Specially, this term is also similar to the energy defined in RBM and Hopfield network.

However, without the probabilistic model in RBM and the recurrent architecture in Hopfield network, maximizing this term merely would not work because the simplified network cannot approximate this assumption and guarantee the stability of solutions without any constraints. Components in W will be $+\infty$ or $-\infty$ when $v^T W h$ is maximized. In order to make the network approximate the assumption and guarantee the long term stability of W and h, a correlation based constraint is introduced. The assumption omits the effect of W on the hidden units h. Therefore, both positive and negative correlations between W and h should be minimized which can be represented by the cosine between the two vectors $\cos^2(h, W_{:j})$ where $W_{:j}$ is the j-th row of the matrix W. Meanwhile, the length of W and h should be limited in order to prevent the infinite increase or decrease of W. Then the correlation constraint can be formulated as the square of inner product between the two vectors $\sum_j \langle h, W_{:j} \rangle^2 = \sum_j \|W_{:j}\|_2^2 \|h\|_2^2 \cos^2(h, W_{:j}) = \|Wh\|_2^2$. With the simplified Hebb model and the correlation based constraint, we can obtain the objective function of NLM:

$$\max J(W, b) = v^T W h - \lambda \|Wh\|_2^2 \tag{3}$$

where λ is a user defined parameter that controls the importance of the two terms. Here we explain the assumption with neuron competition which is inspired from lateral inhibitory.

The network can be taken as undirected graphical model with connections representing the relationship between visible and hidden units as shown in Figure 1. Without competition, the hidden units are independent identically distributed. Because Hebb learning has the property of locality where the update of a weight depends only on the pre- and postsynaptic neurons. The max-clique of this graph is circled by the red ellipse in Figure 1 (a). When the competition is implemented, there are implicit connections between

each pair of hidden units. Then the max-clique includes all the hidden units as shown in Figure 1 (b). Following the probability distribution defined in RBM, we give the energy of the max-clique with the connecting weight $E(C) = \sum_{x_i,x_j \in C} E(x_i, x_j) = -\sum_{x_i,x_j \in C} x_i W_{ij} x_j$ where C denotes the clique and the energy of the clique means the sum of the energy of each connection. The potential of a clique is defined as $\Phi(C) = \exp(-E(C)) = \prod_{x_i,x_j \in C} \Phi(x_i, x_j)$. With the potential, the joint probability density of the graph $p(x_1, x_2, ...) = \prod_C \Phi(C) / \int_{x_1,x_2,...} \prod_C \Phi(C)$. Therefore the distribution of the network in Figure 1 (a) is denoted by:

$$p(v,h) = \frac{\prod_{i,j} \Phi(v_j, h_i)}{\int_{v,h} \prod_{i,j} \Phi(v_j, h_i)} \tag{4}$$

With the simplified architecture, when only Hebb learning is considered, the hidden neurons are independent to each other and all prefer to respond to the most frequent pattern. When the competition is complemented, the potential of a max-clique with v_j is denoted by $\Phi(C_j) = \prod_i \Phi(v_j, h_i) \prod_{k,l} \Phi(h_k, h_l)$. The joint probability of this network is then computed by:

$$\begin{aligned} p_c(v,h) &= \frac{\prod_j \Phi(C_j)}{\int_{v,h} \prod_j \Phi(C_j)} \\ &= \frac{\prod_{i,j} \Phi(v_j, h_i) \left[\prod_{k,l} \Phi(h_k, h_l)\right]^n}{\int_{v,h} \prod_{i,j} \Phi(v_j, h_i) \left[\prod_{k,l} \Phi(h_k, h_l)\right]^n} \end{aligned} \tag{5}$$

where n is the number of visible units. In Eq. (4) and (5), the potential $\Phi(v_j, h_i)$ depends on the status of units as well as W_{ij}. The potential $\Phi(h_k, h_l)$ is then depends on the neuron competition. In neuron competition, the neurons inhibit the firing of other neurons. Therefore, the latent connecting weights between them are usually negative values. Then the summed energy of hidden units $\sum_{kl} E(h_k, h_l) = \sum_{kl} -h_k W_{kl} h_l$ will be a positive large value. The status of the hidden units follows the conditional probability $p(h|v) = p(v,h) / \int_h p(v,h) = \prod_j \Phi(C_j) / \int_h \prod_j \Phi(C_j)$ and therefore with competition, it depends largely on the competition dominated potential while less on the network parameter W. Consequently, we make the assumption in order to model the Hebb rule conveniently.

Since the proposed model has only an encoder, it is convenient to derive a convolutional one in order to learn useful features from images. In convolutional neural networks (CNN), the feature maps are obtained by several trainable convolution kernels:

$$\boldsymbol{H}_i = s(I * \boldsymbol{W}_i + b_i) \tag{6}$$

where $\boldsymbol{H}_i$ denotes the i-th feature map and $\boldsymbol{W}_i$ denotes the i-th convolution kernel. The convolutional NLM (CNLM) can be obtained by summing over all the pixels in feature maps. Then the first term can be denoted by $\sum_{(x,y)} \sum_i (I * \boldsymbol{W}_i)(x,y) \boldsymbol{H}_i(x,y) = \sum_i tr(I * \boldsymbol{W}_i \boldsymbol{H}_i^T)$, where (x,y) denotes the pixel position in $\boldsymbol{H}$ and $tr()$ is the trace of a matrix. For the second term, the convolutional version is also easy to derive, i.e., $\sum_{(x,y)} \sum_{(i,j)} \langle \boldsymbol{W}(i,j), \boldsymbol{H}(x,y) \rangle^2$, where (i,j) denotes the position in convolution kernels. By combining the two terms, the CNLM is formulated:

$$\begin{aligned} \max J_C(\boldsymbol{W}, b) &= \sum_i tr(I * \boldsymbol{W}_i \boldsymbol{H}_i^T) \\ &+ \lambda \sum_{(x,y)} \sum_{(i,j)} \langle \boldsymbol{W}(i,j), \boldsymbol{H}(x,y) \rangle^2 \end{aligned} \tag{7}$$

NLM and CNLM can be optimized by the widely used stochastic gradient descent algorithm.

3 Analysis

In this section, we explain how the NLM learns useful features from input data. As introduced above and described in [Vincent *et al.*, 2010], one natural criterion that we may expect any good representation to meet, at least to some degree, is to retain a significant amount of information about the input. As introduced above, the encoer-decoder and decoder based models can achieve this criterion because of the reconstruction of input data, i.e., $\max p(v|h)$. NLM has only the encoder, but the neuron inspired learning process can also achieve this criterion. In NLM, maximizing Eq. (3) amounts to increasing the cosine correlation coefficient between v and Wh:

$$\cos(v, Wh) = \frac{v^T W h}{\|v\|_2 \|Wh\|_2} \tag{8}$$

Since v is the observed data, maximizing Eq. (3) means to increasing the numerator and decreasing the denominator and therefore increasing the correlation coefficient. Then the conditional probability $p(v|Wh)$ is enlarged. With a distribution of visible data $q(v)$, $\mathbb{E}_{q(v)}[\log p(v|Wh)]$ can be increased. $p(v|Wh)$ is a parametric distribution parameterized by W, therefore it can be written as $p(v|h;W)$. With the deterministic mapping from v to h, $\mathbb{E}_{q(v)}[\log p(v|Wh)]$ can be written as $\mathbb{E}_{q(v,h)}[\log p(v|h)]$. For any distribution $p(v|h)$ we have [Vincent *et al.*, 2010]:

$$\mathbb{E}_{q(v,h)}[\log p(v|h)] \leq \mathbb{E}_{q(v,h)}[\log q(v|h)] \tag{9}$$

$\mathbb{E}_{q(v,h)}[-\log q(v|h)]$ denotes the conditional entropy $\mathbb{H}(v|h)$ between v and h, and the mutual information can be computed by $\mathbb{I}(v;h) = \mathbb{H}(v) - \mathbb{H}(v|h)$ with $\mathbb{H}(v)$ being a constant. Therefore, optimizing the objective function in NLM amounts to increasing the lower bound on the mutual information between v and h. This is similar to AE [Vincent *et al.*, 2010]. Consequently, the representation h learned by NLM can retain information of input data v.

Another criterion of a good representation is to eliminate irrelevant variabilities of the input data, i.e., abstraction and invariance [Bengio *et al.*, 2013]. Many models achieve this criterion by introducing regularization terms, e.g., sparsity induced term [Lee *et al.*, 2007; Ji *et al.*, 2014; Ranzato *et al.*, 2007; Gong *et al.*, 2015] and sensitivity penalization [Rifai *et al.*, 2011]. By introducing noise, the model can be trained to increase the robustness to local variation [Vincent *et al.*, 2010]. This can also learn useful features. In NLM, the abstraction can be achieved by eliminating irrelevant components in v. For each visible unit v_j, the squared

inner product between h and $W_{:j}$ is minimized which breaks the relationship between h and $v_j W_{:j}$ as well. Then the conditional probability $p(h|v_j W_{:j})$ will be minimized. Since $h = f(W, v)$, the conditional probability can be written as $p(h|v_j; W_{:j})$. The constraint term is the sum over all the v_j, and then minimizing the constraint term amounts to minimizing the summed conditional probability $\sum_j p(h|v_j; W_{:j})$. However, maximizing the first term will increase the correlation between $v^T W$ and h which increases the conditional probability $p(h|v; W)$. As a consequence, by maximizing the objective function in NLM, $p(h|v; W)$ is increased while the summed probability over all the components in v, i.e., $\sum_j p(h|v_j; W_{:j})$ is reduced. This means that h can respond to v while h cannot respond to most of the components in v. In other words, in NLM, h responds to most relevant components which omitting the relatively irrelevant components in v. Therefore, the proposed NLM has the ability of abstraction.

From another perspective, in NLM, the first term is similar to the energy defined by RBM:

$$E(v, h) = -\sum_i \sum_j v_j W_{ij} h_i - \sum_i b_i h_i - \sum_j c_j h_j \quad (10)$$

RBM is a generative model and the energy based joint distribution is considered:

$$p(v, h) = \frac{1}{Z} \exp(-E(v, h)) \quad (11)$$

where Z is a normalization constant. RBM is trained by maximizing the probability that the network assigns to the observed data, i.e., $p(v) = \sum_h p(v, h)$. The probability that the network assigns to a training data can be raised by adjusting the weights and biases to lower the energy of that data and to raise the energy of other data, especially those that have low energies and therefore make a big contribution to the partition function [Hinton, 2010]. Similarly, NLM can be taken as an energy based model with a correlation based constraint. Considering negative version of the first term $-v^T W h$ as the energy of the network, the constraint can be used to restrain the decrease of the summed energy of all the data. Since h responds to v according to W, by minimizing the squared inner product between h and $W_{:j}$, the summed energy $\sum_d -d^T W h$ over all the possible data d in the data space is raised to approach to 0 because most components in Wh is restrained to 0. However, since the first term is the energy of the observed data, such energy is relatively lower than that of other data. From the perspective of network energy, the proposed NLM assigns high probability to the observed data which is similar to RBM.

4 Experimental Study

4.1 Experiments on the Learning Process

First we implement the experiments on the gradient based learning process and verify the claims we make in the analysis of the model. The widely used MNIST digit dataset is used here and we randomly select 10000 images from the training set in order to speed up the experiments. We claim

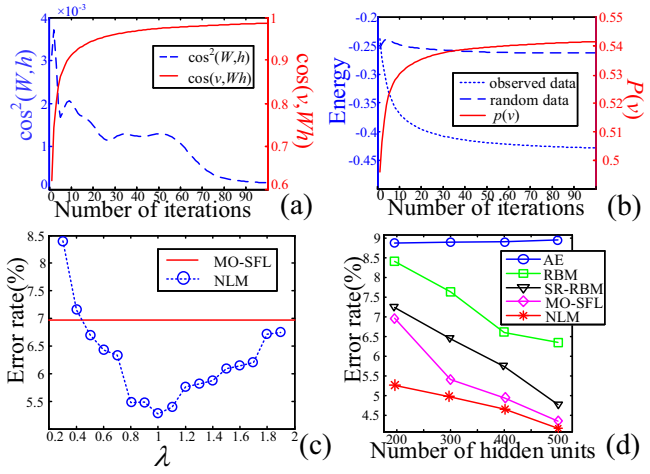

Figure 2: (a) Plots of the correlation coefficient $\cos(v, Wh)$ and $\cos^2(W, h)$ during the learning process. (b) Plots of the energy of the observed data and random data and the simplified $p(v)$ during the learning process. (c) Error rates (%) of NLM with different λ on the MNIST dataset. The red line indicates the error rate of MO-SFL for comparison. (d) Comparison of different learning models with different number of hidden units.

that optimizing the objective function in NLM amounts to increasing the lower bound on the mutual information between v and h. This can be verified by the correlation coefficient between v and Wh in Eq. (8). Therefore we plot the value of $\cos(v, Wh)$ and $\sum_j \cos^2(v_j W_{:j}, h) = \sum_j \cos^2(W_{:j}, h)$ in each iteration as shown in Figure 2 (a). The blue dashed line represents $\sum_j \cos^2(W_{:j}, h)$ and red solid line represents $\cos(v, Wh)$. During the learning process, the correlation between v and Wh increases. This demonstrates that NLM achieves similar function to AE. Then on the squared correlation coefficient between W and h, with the increase of iterations, it gradually converges to 0. This means that the correlation constraint makes the network approximate the assumption and reduce the summed conditional probability $\sum_j p(h|v_j; W_{:j})$.

We also claim that in NLM, h responds to most relevant components which omitting the relatively irrelevant components in v. Then we select two images from the MNIST dataset and CIFAR-10 dataset respectively, and plot $\cos^2(h, W_{:j})$ for each j as the response to each v_j during the iterations. Figure 3 shows such plots of the initialization, 10th iteration (N=10), 50th iteration (N=50) and the last iteration (N=100), respectively. Since v is 2-dimensional data, the plots are exhibited in the 3-dimensional form with the heave representing the response to each pixel in v. The sharp peak demonstrates that h responds to a few components in input data and most components cannot significantly influence the response of h and W. For the image in CIFAR-10 dataset, we also compare the response plots with different λ.

From the network energy perspective, NLM has the similar properties to RBM. We plot the energy $-v^T W h$ in each iteration of the observed data set and other data set as dotted line and dashed line respectively show in Figure 2 (b). The other data set is a random set with the same size to

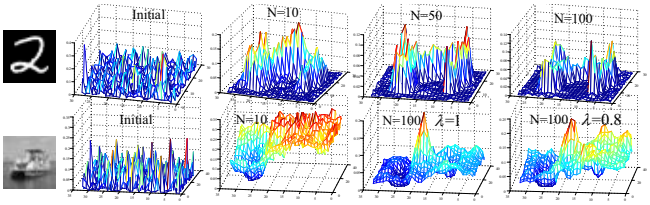

Figure 3: 3 dimension plots of $\cos^2(h, W_{:j})$ on each pixel of the input image. Two images from the MNIST dataset and the CIFAR-10 dataset respectively are exhibited. With the increase of iterations, i.e., N=0, 10, 50, and 100, the irrelevant pixels are restrained. With different λ, the response intensity of each pixel is different.

the observed data set. With the learning process, the energy of observed data decreases greatly while that of the other data does not significantly change. To further narrow the distance between NLM and RBM, we set a simplified $p(v) = \exp(v^T W h)/(\exp(v^T W h) + \exp(d^T W h))$, where d denotes the other data. $p(v)$ in each iteration is plotted into red line as shown in Figure 2 (b). $p(v)$ increases with the learning process which demonstrates that NLM can model the observed data well.

4.2 Experiments on Single-layer Network

Above experiments show the behaviours of NLM during learning process and verify the claims we make in Analysis. Then we test NLM on single-layer network and the features learned by NLM are evaluated. The experiments are implemented on the MNIST digit dataset and image patches. The image patches with the size of 12×12 are randomly extracted from two remote sensing images respectively. Some digit images and the two remote sensing images are shown in Figure 4. The first image is a panchromatic image acquired by Landsat 7 satellite which shows the city of Wuhan in China. The size of the image is 4561×4505 pixels and we randomly extract 100000 image patches as the Wuhan dataset. Another image is a synthetic aperture radar (SAR) image which shows the Yellow River Estuary in China. The size of this image is 3233×5193 pixels and there are 100000 image patches in the Yellow River dataset. Here the remote sensing images are used because of the significance and difficulty of interpreting remote sensing images. Some randomly selected bases learned by NLM on the three datasets are shown in Figure 4. AE, RBM, and the recently proposed sparsity induced version of them, i.e., multiobjective sparse feature learning model (MO-SFL) [Gong *et al.*, 2015] and sparse response RBM (SR-RBM) [Ji *et al.*, 2014] are compared. SR-RBM is based on RBM and use the l_1-norm of the activation probability of hidden units to achieve a small code rate. A user defined parameter is used to control the importance of the two terms. MO-SFL considers the difficulty of determining the compromise between reconstruction error and sparsity. MO-SFL is more adaptive than SR-RBM.

First on the MNIST dataset, without constraints, AE and RBM achieve unstructured weight patterns. Sparsity is a model of simple cells in visual cortex and it has been a key element of unsupervised learning algorithms. With the sparsity constraint, the uncertainty of representations is reduced which increases the robustness to irrelevant variabilities. Therefore, the bases learned by them can capture the local structure of the images as shown in the figures. In NLM, the constraint term leads the representation to responding to most relevant components. Therefore, the local structure can also be captured by the weights. Then on the remote sensing image patches, it is more difficult to deal with remote sensing images than nature images because such images are acquired from satellite and many interferences may influence the quality of the images. As a consequence, AE, RBM and their derivatives may be misled by the inferences. Some bases learned by NLM are Gabor-like edge detectors in different positions and orientations. The results are consistent with many works on nature image patches.

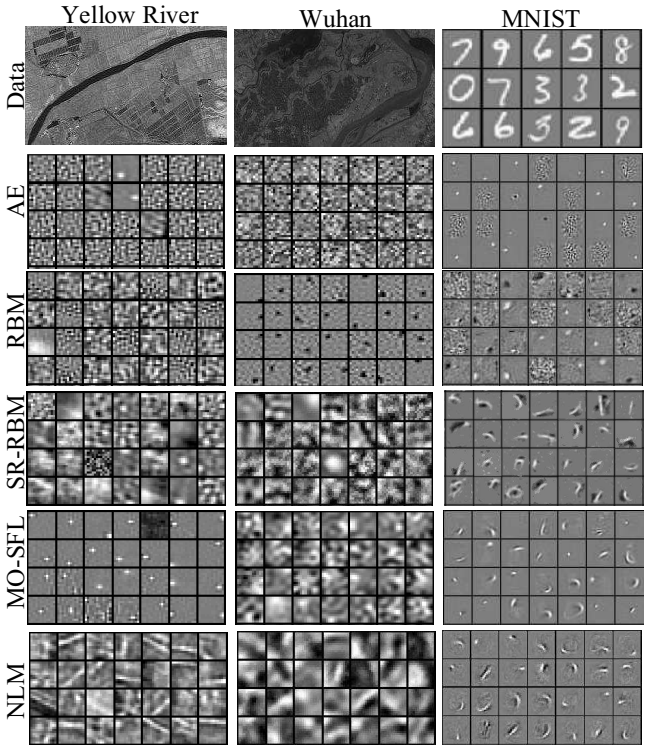

Figure 4: Randomly sampled bases learned by the five models on the MNIST dataset and image patches. The first row exhibits the images in these datasets followed by the bases.

In practice, a good representation is one that will eventually be useful for addressing tasks of interest. Therefore, we test the features learned by the task of image classification. The experiments are implemented on the MNIST dataset and in order to speed up the learning process, we randomly sample 10000 images from the training set to train NLM as well as other models. For the models, we set the number of hidden units to 196. In NLM, $\lambda = 1$ and the learning rate in stochastic gradient descent algorithm is set to 0.01. Those models keep the same number of iterations. The features learned by those models are fed into a linear classifier and we randomly sample 1000, 5000, and 10000 images to train the classifier. The test error rates (%) on the test set are listed in Table 1. The best result is highlighted in bold. From the classification

Table 1: Error rates (%) of features extracted by different feature learning models with single-layer network on the MNIST dataset.

	number of training samples		
models	1000	5000	10000
AE	13.16	10.64	8.88
RBM	12.82	9.70	8.42
SR-RBM	12.49	8.87	7.42
MO-SFL	12.20	8.05	6.97
NLM	**9.98**	**7.10**	**5.27**

test, sparsity induced models can outperform the base models. NLM always achieves the best result.

In NLM, the user defined parameter λ may influence the performance. Therefore, we set $\lambda = 0.03, 0.04, ..., 1.9$ and plot the error rate in Figure 2 (c). The result of MO-SFL is marked by a red line for comparison. We can see that too small or too large values of the λ can reduce the performance of NLM. However, NLM can outperform MO-SFL with a large range of λ values, from 0.5 to 1.9. Meanwhile, relatively good performance can also be achieved with many λ values, from 0.8 to 1.1. Consequently, NLM is not too sensitive to λ.

Another parameter that may influence the performance of learning models is the number of hidden units. In the above experiments, 196 is the empirical value. We set the number of hidden units to 300, 400, and 500 for these learning models and plot the error rate as shown in Figure 2 (d). With the sparse constraint, AE and RBM achieves much better performance when there are more hidden units representing the input data. NLM with 196 hidden units outperforms the other models with 300 hidden units which demonstrates the effectiveness for using hidden units to represent input data.

4.3 Experiments on Deep Networks

Above experiments evaluate NLM on single-layer network. Then we apply NLM to deep neural networks and learn hierarchical representations. First two NLMs are stacked with the hidden units being 500 and 300 respectively. We set $\lambda = 1$ and the learning rate to be 0.01. The compared models, i.e., AE, RBM, SR-RBM, and MO-SFL, are also stacked with the same architecture. The models are trained by the 60000 training images in the MNIST dataset. The learned hierarchical features are fed into a linear classifier and we use randomly sampled 10000 images and 60000 images to train the classifier. The test error rates are listed in Table 2. With the hierarchical representation, NLM can outperform the compared models. However, the performance improvement is much less than single-layer network because of the limited architecture for dealing with images. For efficiently learning features from images, we use the architecture of CNN and learn the features by CNLM.

First on the MNIST dataset, the input of the CNLM is 32×32 images which are obtained by evenly zero padding original images. There are 10 5×5 filters in the first layer and followed by a 2×2 max-pooling layer. The feature maps in the 3rd layer are obtained by 50 $10 \times 5 \times 5$ filters and the 4th layer has 50 down-sampled feature maps by max-pooling. Finally, 200 $50 \times 5 \times 5$ filters are implemented and a feature vector is obtained. Then the features are classified by SVM. We compare CNLM with the convolutional version of AE [Masci *et al.*, 2011] and RBM [Norouzi *et al.*, 2009]. The test error rates are listed in Table 2.

Table 2: Classification results of deep networks including stacked and convolutional version of different feature learning models. In the experiments on convolutional versions, the CIFAR-10 dataset is also used.

	stacked		convolutional	
models	10000	60000	version	CIFAR-10
AE	3.02	1.63	0.76	21.8
RBM	3.06	1.68	0.83	25.2
SR-RBM	2.97	1.61	–	–
MO-SFL	2.91	1.57	–	–
NLM	**2.89**	**1.53**	**0.67**	**21.1**

Then we implement CNLM on a more difficult dataset, i.e., the CIFAR-10 dataset. This dataset consists of 50000 training and 10000 test color images with the size of 32×32. The hierarchical architecture has 5 layers with 3 convolution layers and 2 max-pooling layers, i.e., 3-100-100-150-150-200. The size of filters in convolution layer is 5×5 and the pooling size is 2×2. The learned features are classified by SVM. The convolutional versions of AE and RBM are compared. The test errors are listed in Table 2. Experiments on deep neural networks demonstrate the effectiveness of NLM for learning hierarchical features.

5 Conclusions

In this paper, a neuron learning machine (NLM) was proposed based on Hebb learning. Hebb learning rule describes how synaptic weight changes and can be formulated by the gradient of a connecting weight in neural network models. To model the network with Hebb learning, we integrate over the gradient by assuming the negligible effect of weights on representations. This omitting assumption simplifies the integral process greatly. In order to approximate this assumption and obtain a stable solution, a correlation constraint is added. We find that NLM has the properties of retaining information and abstraction which are essential for learning useful representations. Analysis and experiments demonstrate the effectiveness of the learned features by the proposed model. The future work will include but not limited to exploring the learning behaviours of neurons for efficient unsupervised as well as supervised neural learning models on various network architectures.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61422209, 61672409), and the National Program for Support of Top-notch Young Professionals of China.

References

[Arel *et al.*, 2010] Itamar Arel, Derek Rose, and Thomas Karnowski. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE Computational Intelligence Magazine*, 5(11):13–18, 2010.

[Bengio *et al.*, 2006] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proc. NIPS*, 2006.

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. PAMI*, 35(8):1798–1828, 2013.

[Gong *et al.*, 2015] Mao Guo Gong, Jia Liu, Hao Li, Qing Cai, and Lin Zhi Su. A multiobjective sparse feature learning model for deep neural networks. *IEEE Trans. NNLS*, 26(12):3263–3277, 2015.

[He *et al.*, 2013] Yun Long He, Koray Kavukcuoglu, Yun Wang, Arthur Szlam, and Yan Jun Qi. Unsupervised feature learning by deep sparse coding. *arXiv:1312.5783*, 2013.

[Hebb, 1949] Donald Olding Hebb. *In The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, 1949.

[Hinton and Salakhutdinov, 2006] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Hinton *et al.*, 2006] Geoffrey Hinton, Simon Osindero, and YeeWhye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[Hinton, 2010] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):599–619, 2010.

[Ji *et al.*, 2014] Nan Nan Ji, Jiang She Zhang, and Chun Xia Zhang. A sparse-response deep belief network based on rate distortion theory. *Pattern Recognition*, 47(9):3179–3191, 2014.

[Kruger *et al.*, 2013] Norbert Kruger, Peter Janssen, Sinan Kalkan, Markus Lappe, Ales Leonardis, Justus Piater, Antonio Jose Rodriguez-Sanchez, and Laurenz Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Trans. PAMI*, 35(8):1847–1871, 2013.

[Kuriscak *et al.*, 2015] Eduard Kuriscak, Petr Marsalek, Julius Stroffek, and Peter Toth. Biological context of Hebb learning in artificial neural networks, a review. *Neurocomputing*, 152:27–35, 2015.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffery Hinton. Deep learning. *Nature*, 521(7553):436–44, 2015.

[Lee *et al.*, 2007] Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. Sparse deep belief net model for visual area V2. In *Proc. NIPS*, 2007.

[Masci *et al.*, 2011] Jonathan Masci, Ueli Meier, Ciresan Dan, and Jrgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proc. ICANN*, pages 52–59, 2011.

[Norouzi *et al.*, 2009] Mohammad Norouzi, Mani Ranjbar, and Greg Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *Proc. CVPR*, pages 2735–2742, 2009.

[Ranzato *et al.*, 2007] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *Proc. NIPS*, 2007.

[Rifai *et al.*, 2011] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. ICML*, 2011.

[Scellier and Bengio, 2016] Benjamin Scellier and Yoshua Bengio. Towards a biologically plausible backprop. *arXiv:1602.05179*, 2016.

[Valpola, 2014] Harri Valpola. From neural PCA to deep unsupervised learning. *arXiv:1411.7783*, 2014.

[Vincent *et al.*, 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.