Unsupervised Training Algorithm for Recirculation Neural Network

Vladimir Golovko, Vitaly Gladyschuk

Department of Computers, Brest polytechnic institute, Moscowskaja 267, 224017 Brest, Republic of Belarus, ph: +375 162 421081, fax: +375 162 422127, e-mail: cm@brpi.belpak.brest.by

Abstract

Unsupervised learning is the great promise of the future. In such training the network is provided with inputs but not with desired outputs. Unsupervised learning is used for the principal component networks. This paper describes a new method for training of the recirculation networks. Such method is called a sectioning learning. It is characterized by small training time and stability of training.

Keywords. Recirculation neural network, unsupervised training, compression.

1. Introduction

As it is known from statistics[1] the Principal Component Analysis (PCA) is the basic tool to reduce dimension by eliminating redundant variables. Such transformation from the n-dimensional to the m-dimensional vector equals the concatenation of the first m eigenvector of the correlation matrix of the input signals. In this procedure the input space is rotated in such a way that the output values are as uncorrelated as possible and the energy or variances of the data is mainly concentrated in a few first principal components.

The principal component networks (recirculation networks) use the various variants of unsupervised learning. So many papers are based on Hebbian learning [2,3,4]. Such learning rule is the equivalent to the Principal Component Analyses. Other authors [5,6,7] have used the backprogation algorithm or cumulative delta rule. However these algorithms have excessive training times and lack of convergence to an acceptable solution. There is no guarantee that the learning will be a success.

This paper describes a new method for training the recirculation network. Such method is characterized by small training times and stability of training. Various numerical experiments are used to illustrate the potential of the suggested method.

2. Architecture

Recirculation networks are characterized both feed-forward $y = f(x)$ and feed-back $\bar{x} = f(y)$ transformation of the patterns. Such networks are used for compression (feed-forward transformation) and decompression (feed-back transformation) of the data. The architecture of the recirculation neural network is shown on Fig.1. It consists of three layers. The input units receive data from outside and distribute these data to hidden units. The hidden units perform the compression of the input data X:

$$Y = F(WX) \tag{1}$$

The units in the output layer are meant for decompression of the data of the hidden layer:

$$\bar{X} = F(W'Y), \tag{2}$$

where W is the weight matrix; X is the input vector; Y is the output vector of the hidden layer; $\bar{X}$ is the decompressed vector; F is the activation function.

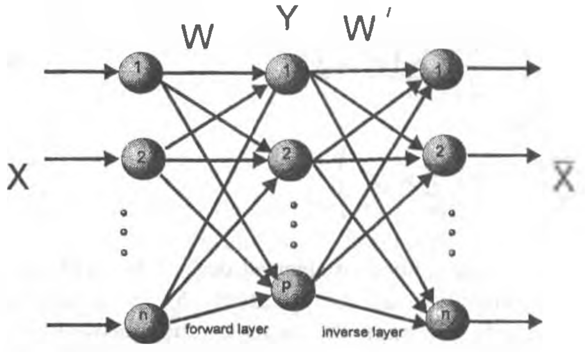

Figure 1.

One can see in Fig.1, that recirculation network has two weight layers, called the forward and inverse layers. The purpose of the learning is to reduce the error between the decompressed vector and the input vector.

3. Background and motivation

The proposed method involves two separate phases of the training. At the first phase the definition of the weight matrix W' and principal components $\bar{Y}$ for minimization of the total square error between the decompressed data and the original data set is performed. For this purpose it is necessary to minimize the following equation:

$$|X - W'Y| \rightarrow \min \tag{3}$$

It is equivalent to the minimization

$$E_t = \frac{1}{2}\sum_{k=1}^{L}\sum_{i=1}^{n}(\bar{x}_i^k - x_i^k)^2, \tag{4}$$

where L is the quantity of the input data.
By this

$$\bar{x}_i = F\left(\sum_{j=1}^{p} w'_{ji} y_j\right), \tag{5}$$

where p is the dimension of the hidden layer and $p < n$. For the minimization of the equation (5) the method of steepest descent can be used. As a result the weights w'_{ij} of inverse layer and principal components y_j are defined.

At the second phase the weight matrix W of the forward layer is determined. As the target outputs the vector $\bar{Y}$ is used, which has been obtained previously. Then the aim of the training at the second phase is to minimize the following expression:

$$E'_t = \frac{1}{2}\sum_{k=1}^{L}\sum_{i=1}^{n}(y_i^k - \bar{y}_i^k)^2, \tag{6}$$

where

$$y_j = F\left(\sum_{i=1}^{n} w_{ij} x_i\right) \tag{7}$$

The method of steepest descent we will use for minimization of the equation (6). Such approach permits to train the inverse and forward layers separately. Let's consider the training rules for linear and nonlinear networks.

4. Linear network

Such networks use a linear activation function and perform the linear compression of a data set. Then the outputs of the forward layer are as follows

$$y_j = \sum w_{ij} x_i, \tag{8}$$

where $j = \overline{1, p}$.
The outputs of the inverse layer are given by

$$\bar{x}_i = \sum_{i=1}^{p} w'_{ji} y_j, \tag{9}$$

where $i = \overline{1, n}$.
The learning problem at the first phase can be formulated as: how do we compute $\Delta w'_{ij}(t)$ and $\Delta y_j(t)$ in order to minimize the total mean-square error between the decompressed data $\bar{X}$ and the original data set X (equation 4)? Let's examine the definition of the vector $\bar{Y}$ for the minimization of the equation (4). For this purpose the gradient descent method is used. Then the learning rule is

$$y_j(t+1) = y_j(t) - \alpha_j(t)\frac{\partial E}{\partial y_j(t)}, \tag{10}$$

where $\alpha_j(t)$ is the adaptive training rate for neuron j of the hidden layer.
The error E is defined by the expression:

$$E = \frac{1}{2}\sum_{i=1}^{n}(\bar{x}_i - x_i)^2 \tag{11}$$

Then the error derivate is

$$\gamma_j = \frac{\partial E}{\partial y_j} = \frac{\partial E}{\partial x_i}\frac{\partial \bar{x}_i}{\partial y_j} = \sum_i w'_{ji}(\bar{x}_i - x_i) \tag{12}$$

Theorem 1. The adaptive training rate $\alpha_j(t)$ is defined by the following expression:

$$\alpha_j(t) = \frac{1}{\sum_i (w'_{ji}(t))^2} \tag{13}$$

Proof. In order to compute the training rate $\alpha_j(t)$ we will use the method of the steepest descent. From this follows that

$$\alpha_j(t) = \min\left\{E\left(y_j(t) - \alpha_j(t)\frac{\partial E}{\partial y_j(t)}\right)\right\}$$

The activation values of element i can be written as:

$$\bar{x}_i(t+1) = w'_{ji}(y_j(t) - \alpha_j \gamma_j) + \sum_{k \neq j} w_{ki} y_k(t)$$

After modifications we have

$$\bar{x}_i(t+1) = \bar{x}_i(t) - \alpha_j \gamma_j w'_{ji}$$

The error function for a pattern

$$E = \frac{1}{2} \sum_{i=1}^{n} (\bar{x}_i(t+1) - x_i)^2$$

Then the derivate is

$$\frac{\partial E}{\partial \alpha_j} = \sum_i (\bar{x}_i(t) - x_i - \alpha_j \gamma_j w'_{ji}) * (-\gamma_j w'_{ji}) = 0 \quad (14)$$

Now we can determine the adaptive rate.

From equation (14) follows that:

$$\alpha_j = \frac{\gamma_j}{\sum_{i=1}^{n} (w'_{ji}(t))^2} \quad (15)$$

It may be noted, that

$$\frac{\partial^2 E}{\partial \alpha_j^2} > 0$$

From this follows that the equation (15) minimizes the error function. Thus the learning rate is adapted to connection weights w'_{ji}.

The learning rule is

$$y_j(t+1) = y_j(t) - \frac{\gamma_j}{\sum_{i=1}^{n} (w'_{ji}(t))^2} \quad (16)$$

Now we have to determine the learning rule for connection weights w'_{ji}. In accordance to the gradient descent method

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha(t) \frac{\partial E}{\partial w'_{ji}(t)} \quad (17)$$

where $\alpha(t)$ is adaptive learning rate for inverse layer.

In this case we can get that

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha(t)(\bar{x}_i - x_i) y_j. \quad (18)$$

The learning rate can be defined by analogy with theorem 1. Then

$$\alpha(t) = \frac{1}{\sum_{j=1}^{p} y_j^2(t)} \quad (19)$$

As can be seen, the learning rate is adapted to every pattern Y.

At the second phase it is necessary to compute $\Delta w_{ij}(t)$ in order to minimize the total mean-square error between the data Y and the target data set $\bar{Y}$ (equation 6). Then we have that

$$w_{ij}(t+1) = w_{ij}(t) - \alpha(t) \frac{\partial E'}{\partial w_{ij}(t)}. \quad (20)$$

The error function for a pattern

$$E' = \frac{1}{2} \sum_{j=1}^{n} (y_j - \bar{y}_j)^2 \quad (21)$$

where $\bar{y}_j$, $j = \overline{1,p}$, are the first principal components.

By using the gradient descent method we can get that

$$w_{ij}(t+1) = w_{ij}(t) - \alpha(t)(y_j - \bar{y}_j) x_i \quad (22)$$

The learning rate is

$$\alpha(t) = \frac{1}{\sum_{j=1}^{n} x_j^2(t)}. \quad (23)$$

5. Nonlinear networks

The nonlinear activation function is used for such network. By using nonlinearity of better results can be achieved in comparison with the linear function. Let's consider the training rules for nonlinear networks. As the activation function we will use a hyperbolic tangent. Then the j th output of a forward layer is given by

$$y_j = th(s_j), \quad (24)$$

where $s_j = \sum_{i=1}^{n} w_{ij} x_i$, (25)

where $j = \overline{1,p}$.

Accordingly for an inverse layer

$$\bar{x}_i = th(s_i) \quad (26)$$

$$s_i = \sum_{j=1}^{p} w'_{ji} y_j \quad (27)$$

Let's examine the training rules. At the first phase it is necessary for each input vector x to define such vector y, which will ensure the minimization of expression (4).

The error derivate is equal to

$$\gamma_j = \frac{\partial E}{\partial y_j} = \frac{\partial E}{\partial x_i} \frac{\partial \bar{x}_i}{\partial s_i} \frac{\partial s_i}{\partial y_j} = \sum_i (\bar{x}_i - x_i)(1 - \bar{x}_i^2) w'_{ji} \quad (28)$$

Then

$$y_j(t+1) = y_j(t) - \alpha_j(t)\sum_i (\bar{x}_i - x_i)(1-\bar{x}_i^2)w'_{ji} \cdot \quad (29)$$

By using Taylor series decomposition and the steepest descent method we can receive the adaptive training rate:

$$\alpha_j = \frac{\sum_{i=1}^{n}(\bar{x}_i - x_i)w'_{ji}}{\gamma_i \sum_{i=1}^{n}(w'_{ji})^2}. \quad (30)$$

Let's define the expression for the modification of the weights W'. Then:

$$\frac{\partial E}{\partial w'_{ji}} = \frac{\partial E}{\partial \bar{x}_i}\frac{\partial \bar{x}_i}{\partial S_i}\frac{\partial S_i}{\partial w'_{ji}} = (\bar{x}_i - x_i)(1-\bar{x}_i^2)y_j \quad (31)$$

By using Taylor series decomposition and the steepest descent method we can receive the adaptive training rate:

$$\alpha(t) = \frac{\sum_{i=1}^{n}(\bar{x}_i - x_i)^2(1-\bar{x}_i^2)}{(\sum_{j=1}^{p} y_j^2)\sum_{i=1}^{n}(\bar{x}_i - x_i)^2(1-\bar{x}_i^2)^2}. \quad (32)$$

Then the modification of weight connections W' is as follows:

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha(t)(\bar{x}_i - x_i)(1-\bar{x}_i^2)y_j. \quad (33)$$

Thus at the first stage of the training the weights of the inverse layer w'_{ji} and p - first principal components $\bar{y}_j$ are calculated.

At the second phase the weights of the forward layer are calculated, where the values Y are used as target outputs. For this purpose it is necessary to minimize the mean-square error (equation 6).

By using the gradient descent method we can get:

$$\frac{\partial E'}{\partial w_{ij}} = \frac{\partial E'}{\partial y_j}\frac{\partial y_j}{\partial S_j}\frac{\partial S_j}{\partial w'_{ij}} = (y_j - \bar{y}_j)(1-y_j^2)x_i, \quad (34)$$

By using Taylor series decomposition and the steepest descent method we can get the adaptive training rate:

$$\alpha(t) = \frac{\sum_{j=1}^{p}(y_j - \bar{y}_j)^2(1-y_j^2)}{(\sum_{i=1}^{n} x_i^2)\sum_{j=1}^{p}(y_j - \bar{y}_j)^2(1-y_j^2)^2}. \quad (35)$$

Then

$$w_{ij}(t+1) = w_{ij}(t) - \alpha(t)(y_j - \bar{y}_j)(1-y_j^2)x_i. \quad (36)$$

6. Proposed algorithm

Let L be the quantity of input patterns. The algorithm of sectioning training consist of the following steps:

1. Randomly initialization of weights. Choice of the minimum total mean-square error E_m.

2. Cosequently L of patterns enter the neural network inputs. By this for each pattern only a feed-forward transformation of data is performed. As a result of the given stage the set of the compressed images Y is defined which will be used at the next stage of the algorithm.

3. Only the inverse layer is considered. As the input information the values Y are used which are defined on the previous step of algorithm. As the target patterns of the inverse layer the vector of input data X is used. The following sequence of operations is performed:

   3.1. For L patterns Y weights update using the expression (18) for linear or (33) for nonlinear neural networks.

   3.2. For L patterns Y outputs update of the hidden layer using expression (16) for linear or (29) nonlinear neural networks.

   3.3. The steps 3.1 and 3.2 are repeated, until the total mean-square error of an inverse layer becomes smaller than the given E_m.

4. The modification of the weight of the forward layer is performed according to expression (22) for linear or (36) nonlinear neural networks. For this purpose the input patterns consequently enter at the network and there is only forward transformation of the information for each pattern. The values $\bar{Y}$ obtained on the previous step of algorithm are used as the target data.

5 The step 4 proceeds untill the total mean-square error of the forward layer becomes smaller than the given one (E_m).

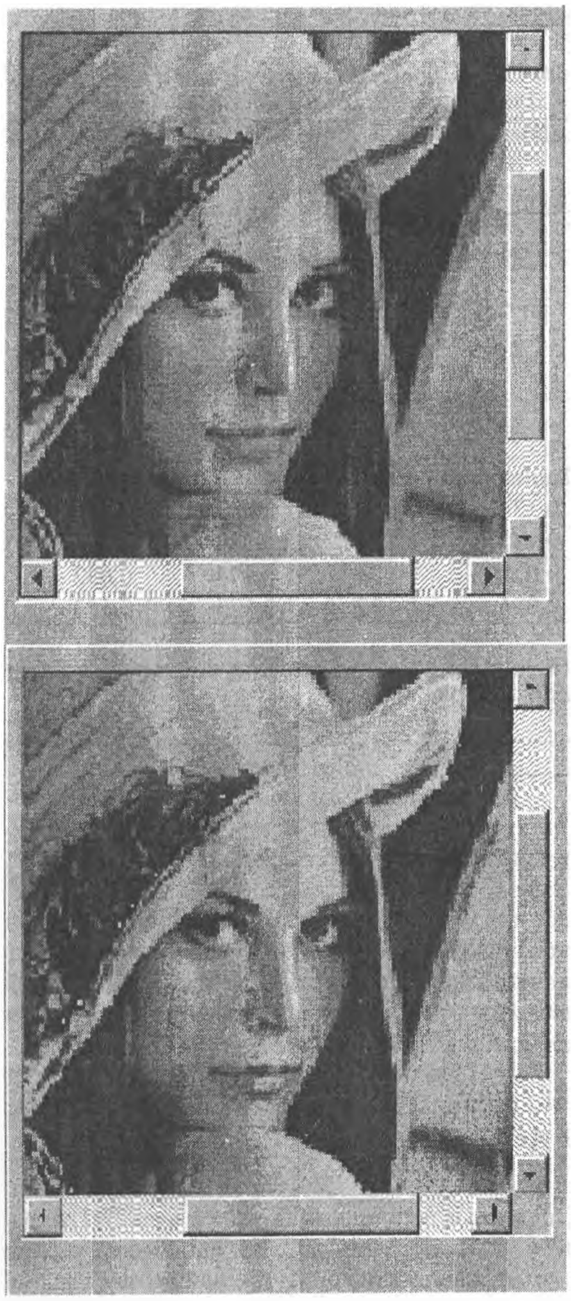
Figure 2. Fragments of the test image(at the top) and decompressed image (at the down).

For definition of weights the normalized training rule can be used.

By using this algorithm it is possible to train the nonlinear recirculation network for the compression of the data. The experiments have shown that the offered algorithm is more effective in comparison with the usual ones [5-7].

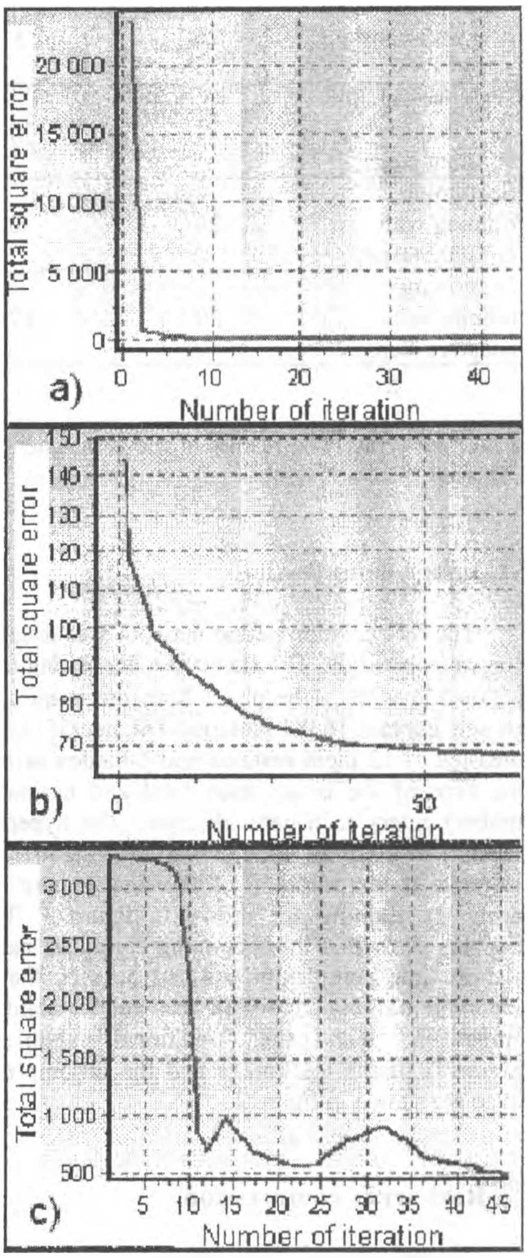

Figure 3. Diagrams of the variation of the total square error during training:
a) sectioning training; b) back-propagation algorithm; c) cumulative delta rule.

7. Experiments

We have applied the proposed algorithm to the task of data compression. In the first experiment we illustrate the application of our results for the compression of human faces. In the second experiment our algorithm is used for the compression

Table 3. Comparison of different training methods.

Training method	Number of tests	Number of successful tests	Number of failure[1] tests	Average number of iterations	Average time of training, sec.
Cumultative delta rule with constant steps	20	3	17	2,07E4	27,4
Back propagation algorithms with constant steps	20	14	6	3,09E3	41,2
Sectioning training with constant steps	20	20	20	899	1,7
Sectioning training with adaptive steps	20	17	3	19,2	0,0412

of the real array. The results of the experiments are discussed.

7.1. Image compression

The recirculation neural network was tested for image compression. The standard color testing image is 256×256 with 24 bit/pixel. Both the training and test sets contain 16384 patterns. The neural network consisted of 12 input neurons and 6 hidden neurons. One byte of the information (real and normalized number) enters each network input. The hyperbolic tangent is used as the activation function. The diagrams of the variation of the total square error during the training are shown in figure 3. These diagrams show that the sectioning algorithm has the smallest time complexity. As can be seen that the sectioning training algorithm has good stability in comparison with the traditional algorithms. Fragments of the test image and the decompressed image are shown in figure 2.

7.2. Real array compression

For the analysis of the abilities of our training method we tested recirculation neural network for compression of the real arrays. In this tests we used different real arrays as the input data. The example of one of them is presented in table 1. Table 2 shows the compressed array after neural network training. In this case we use neural network with 4 input neural elements and 2 hidden ones. Also the hyperbolic tangent was used as the activation function. The total main square error for a successful test must be less than 0,01. Table 3 shows information about different training methods for compression of the given real array.

Table1. Input array.

0,134	0,045	0,032	0,032
0,135	0,134	0,032	0,023
0,072	0,073	0,144	0,032
0,025	0,125	0,123	0,123

Table 2. Compressed array for sectioning training with adaptive steps.

0,211	0,613
0,332	0,8

8. Conclusion

Our sectioning algorithm is an efficient tool for reducing of the dimension of the data. This algorithm will be used for training of the recirculation neural networks. We are currently exploring other applications of our approach.

9. Acknowledgments

This work is performed within the project INTAS 97-0606 “Development of an intelligent sensing instrumentation structure”. The authors are thankful to the European Community for the financial support of the project.

10. References

1. Jolliffe I.T. Principal Component Analysis. // Springer-Verlag. 1986.
2. E. Oja, H. Ogawa, and J. Wangviwattana. Pca in fully parallel neural networks. In Aleksander

[1] if a total main square error reduced less than 1E-5 for 100 iteration.

&Taylor, editor, Artificial Neural Networks,2, 1992.

3. T.D. Sanger. Analysis of the two-dimensional receptive fields learned by the generalized hebbian algorithm in response to random input. Biological Cybernetics, 1990.
4. C. Fyfe and R. Baddeley. Non-linear data structure extraction using simple hebbian networks. Biological Cybernetics, 72(6):533{541, 1995.
5. Cottrell G., Munro P., Zipser D. Image compression by back-propagation: a demonstration of extensional programming // Tech. Rep. N.TR8702.–USCD: Institute of Cognitive Sciences –1987
6. Hinton G., McClelland J. Learning Representation by Recirculation // Proceedings of IEEE Conference on Neural Information Processing Systems.–1989.
7. Cottrell G., Munro P., Zipser D. Learning Internal Representation from Gray–Scale Images: An Example of Extensional Programming // Proceedings 9th Annual Conference of the Cognitive Science Society – 1987.–P.461–473.