computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

SOME METHODS OF ADAPTIVE MULTILAYER NEURAL NETWORKS TRAINING

Leonid Makhnist, Nikolaj Maniakov, Vladimir Rubanov

Brest State Technical University, Department of High Mathematics,
Moskovskaja 267, 22417, Brest, Republic of Belarus

Abstract: *Is proposed two new techniques for multilayer neural networks training. Its basic concept is based on the gradient descent method. For every methodic are showed formulas for calculation of the adaptive training steps. Presented matrix algorithmizations for all of these techniques are very helpful in its program realization.*

Keywords: *Multilayer Neural Networks, Gradient Descent Method, Adaptive Training Step.*

1. INTRODUCTION

Let examine multilayer neural network, consisting of N neural blocks (Fig.1). Each of these blocks has a structure described in Fig. 2.

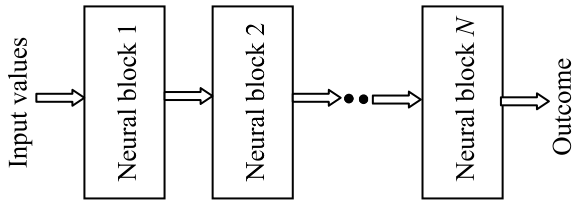


Fig.1 – Multilayer neural network

Output values of each neural block are input values for the next block; input values for the first block are sequence of input patterns $\overline{x^k}=\left(x_1^k,...,x_{m_0}^k\right)^T$, $\left(k=\overline{1,L}\right)$. Output value of i_n-th neuron of n-th block for a k-th pattern is defined by recurring expression

$$y_{i_n}^{(n),k}=F_n\left(S_{i_n}^{(n),k}\right),$$

where

$$S_{i_n}^{(n),k}=\sum_{i_{n-1}=1}^{m_{n-1}} w_{i_{n-1}i_n}^{(n)} y_{i_{n-1}}^{(n-1),k}-T_{i_n}^{(n)},\quad i_n=\overline{1,m_n},\quad k=\overline{1,L}.$$

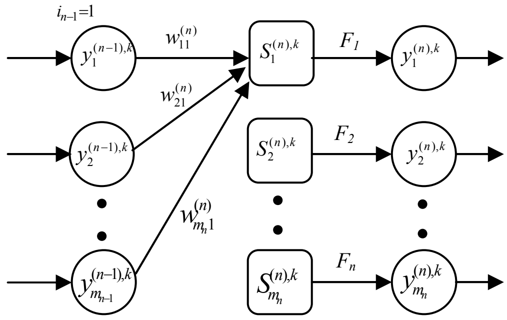


Fig.2 – Architecture of *n* neural block

So we form a vector

$$Y^{(n),k}=\left(y_1^{(n),k}\quad y_2^{(n),k}\quad \dots\quad y_{m_n}^{(n),k}\quad -1\right)^T.$$

The task of such neural networks' training consist in finding of weights' coefficients matrix

$$W^{(n)}=\begin{pmatrix} w_{11}^{(n)} & w_{21}^{(n)} & \dots & w_{m_{n-1}1}^{(n)} \\ w_{12}^{(n)} & w_{22}^{(n)} & \dots & w_{m_{n-1}2}^{(n)} \\ \dots & \dots & \dots & \dots \\ w_{1m_n}^{(n)} & w_{2m_n}^{(n)} & \dots & w_{m_{n-1}m_n}^{(n)} \end{pmatrix}_{m_n\times m_{n-1}}$$

and vectors of thresholds $\overline{T^{(n)}}=\left(T_1^{(n)},\ T_2^{(n)},...,\ T_{m_n}^{(n)}\right)^T$, $n=\overline{1,N}$, which minimize some network error E_S. That error characterizes deviation of network outcome values $y_{i_N}^{(N),k}$ from standard $t_{i_N}^k$ for each i_N-th neural element for k-th pattern. We take a mean-square error as criterion function:

$$E_S=\frac{1}{2L}\sum_{k=1}^{L}\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)^2.$$

2. TRAINING ALGORITHM

For a program realization of such neural networks' training process is very helpful its matrix algorithmization [1], described by the next way:

Modifications of synaptic connection in multilayer heterogeneous neural network are produced accordance to the formulas:

$$w_{j_{n-1}j_n}^{(n)}\left(t+1\right)=w_{j_{n-1}j_n}^{(n)}\left(t\right)-\alpha^{(n)}\cdot\frac{1}{L}\cdot\sum_{k=1}^{L}C^{(n)}\cdot M_{j_nj_{n-1}}^{(n)}\cdot Y^{(n-1),k}$$

$$T_{j_n}^{(n)}\left(t+1\right)=T_{j_n}^{(n)}\left(t\right)-\alpha^{(n)}\cdot\frac{1}{L}\cdot\sum_{k=1}^{L}C^{(n)}\cdot M_{j_n(m_{n-1}+1)}^{(n)}\cdot Y^{(n-1),k}$$

where $C^{(n)}$ is calculated recurrently:

$$C^{(n)}=C^{(n+1)}\cdot W^{(n+1)}\cdot MF_n \quad , \quad C^{(N)}=\varepsilon^k\cdot MF_N$$

$$\varepsilon^k=\left(\left(y_1^{(2),k}-t_1^k\right)\ \left(y_2^{(2),k}-t_2^k\right)\ \ldots\ \left(y_{m_2}^{(2),k}-t_{m_2}^k\right)\right),$$

and $MF_n=\begin{pmatrix} F_n'\left(S_1^{(n),k}\right) & 0 & \ldots & 0 \\ 0 & F_n'\left(S_2^{(n),k}\right) & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & F_n'\left(S_{m_n}^{(n),k}\right) \end{pmatrix}$ - are $m_n\times m_n$ matrixes, $M_{j_n j_{n-1}}^{(n)}$ - are $m_n\times\left(m_{n-1}+1\right)$ matrixes consisting of zero elements with only element in position $j_n j_{n-1}$, which value is equal to one.

Synaptic connection changes begin from the last layer down to first.

Using such training methodic we can take a training step $\alpha^{(n)}$ like a constant or a adaptive. Last case is more effective. For a twolayer neural network we can take it in accordance to the one of the next method: layerwise training, two-parameter training and generalized method of fastest descent [2]. But spreading some of them in to the neural networks with more then two layers architecture gives very complicated formulas. So we proposed the next two methods, which basic principals deal with matrix algorithmization and fastest descent method.

3. TRAINING ALGORITHM BASED ON THE NETWORKS' ERROR CONDITIONAL OPTIMIZATION

We proposed new heuristic method of neural networks' training process with use of adaptive training step. Such method based on conditional minimization of the each layers' error. By use of this method we consider each layer like onelayer neural network, which training produced by gradient descent method. And we aimed output of each layer to the received "standard". So, we must recalculate "standard" values through all training process.

The algorithm of thus method can be described in the next way:

Procedure Network training
begin
  set training accuracy ε
  repeat
    modification of N layer synaptic connection
    for n=N-1 **down to** 1 **do**
    begin
      for k=1 **to** L **do**
        begin
          finding of "standard" output of n-th layer for each pattern
        end
      modification of n-th layer synaptic connection
    end
    finding the training error E_S
  until $E_S<\varepsilon$
end.

This algorithm is based on the next theorem.

Theorem. By using of above algorithm we must calculate "standard" output values accordance to the formulas

$$t_{j_n}^{(n),k}=y_{j_n}^{(n),k}-\alpha\cdot C^{(n+1)}\cdot W^{(n+1)}\cdot\Delta_{j_n}^{n},\ j_n=\overline{1,m_n},$$

with the next correction :

$$t_{i_n}^{(n),k}:=\begin{cases} a+\beta, & if\ \ t_{i_n}^{(n),k}<a+\beta \\ t_{i_n}^{(n),k}, & if\ \ t_{i_n}^{(n),k}\in\left[a+\beta,b-\beta\right], \\ b-\beta, & if\ \ t_{i_n}^{(n),k}>b-\beta \end{cases}$$

where parameter β must be taken by us as a small number.

Adaptive training step can by taken in the next way:

$$\alpha=\frac{\sum_{j_n=1}^{m_n}\left(C^{(n+1)}\cdot P_{j_n}^{n}\right)^2}{\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\left(C^{(n+1)}\cdot P_{j_n}^{n}\right)\cdot\left(\left(P_{j_n}^{n}\right)^T\cdot U^{(n+1),k}\cdot\left(P_{l_n}^{n}\right)\right)\cdot\left(C^{(n+1)}\cdot P_{l_n}^{n}\right)},$$

where

$$U^{(n),k}=\left(W^{(n+1)}\cdot MF_n'\right)^T\cdot U^{(n+1),k}\cdot\left(W^{(n+1)}\cdot MF_n'\right)+W^{(n+1)}\cdot MF_n''$$

$$U^{(N),k}=\left(MF_N'\right)^2+DE^{(N),k}\cdot MF_n',$$

$$P_{j_n}^{n}=W^{(n+1)}\cdot\Delta_{j_n}^{n},$$

$$DE^{(N),k}=diag\left(\left(y_1^{(N),k}-t_1^k\right)\ \left(y_2^{(N),k}-t_2^k\right)\ \ldots\ \left(y_{m_2}^{(N),k}-t_{m_2}^k\right)\right)$$

and $\Delta_{j_n}^{n}$ - zero vector-column with one element in a position j_n equal to 1.

Modification of weights and threshold is produced accordance to

$$w_{j_{n-1}j_n}^{(n)}\left(t+1\right)=w_{j_{n-1}j_n}^{(n)}\left(t\right)-\alpha^{(n)}\cdot G_{j_{n-1}j_n}^{(n),layer},$$

$$T_{j_n}^{(n)}\left(t+1\right)=T_{j_n}^{(n)}\left(t\right)-\alpha^{(n)}\cdot G_{j_{n-1}j_n}^{(n),layer}$$

for $j_{n-1}=\overline{1,m_{n-1}}$, $j_n=\overline{1,m_n}$, with adaptive training step

$$\alpha^{(n)}=\frac{\sum_{j_{n-1}=1}^{m_{n-1}+1}\sum_{j_n=1}^{m_n}\left(G_{j_{n-1}j_n}^{(n),layer}\right)^2}{\sum_{j_{n-1},l_{n-1}=1}^{m_{n-1}+1}\sum_{j_n,l_n=1}^{m_n}G_{l_{n-1}l_n}^{(n),layer}\cdot\left(S_{(n)}\right)_{l_{n-1}l_l}^{j_{n-1}j_n}\cdot G_{l_{n-1}l_n}^{(n),layer}},$$

where $G_{j_{n-1}j_n}^{(n),layer}=\sum_{k=1}^{L}C_{layer}^{(n),k}\cdot K_{j_{n-1}j_n}^{(n),k}$, $C_{layer}^{(n),k}=\varepsilon_n^k\cdot MF_n'$,

$K_{ij}^{(n),k}=M_{ji}^{(n)}\cdot Y^{(n-1),k}$,

and

$$\left(S_{(n)}\right)_{l_{n-1}l_l}^{j_{n-1}j_n} = \sum_{k=1}^{L}\left(\left(K_{l_{n-1}l_n}^{(n),k}\right)^T \cdot \left(\left(MF_n'\right)^2 + DE^{(n),k} \cdot MF_n''\right) \cdot K_{j_{n-1}j_n}^{(n),k}\right)$$

$$K_{j_{n-1}j_n}^{(n),k} = M_{j_nj_{n-1}}^{(n)} \cdot Y^{(n-1),k}.$$

Proof. Let us examine n-th block of our multilayer neural network (Fig. 2). We will consider it as onelayer feedforward neural network with input values $Y^{(n-1),k} = \left(y_1^{(n-1),k} \quad y_2^{(n-1),k} \quad \dots \quad y_{m_{n-1}}^{(n-1),k} \quad -1\right)^T$ and output described as follows.

The process of finding "standard" values $t_{i_n}^{(n),k}$, $i_n = \overline{1,m_n}$ of outputs in n-th neural layer on the basis of gradient descent method has the next form:

$$t_{j_n}^{(n),k} = y_{j_n}^{(n),k} - \alpha \cdot \frac{\partial E_s^k}{\partial y_{j_n}^{(n),k}},\ j_n = \overline{1,m_n}.$$

Based on these formulas we denote finding values $y_{j_n}^{(n),k}(t+1)$ as a "standard" $t_{i_n}^{(n),k}$ for the next modification of synaptic connection in n-th layer.

Let's find the partial derivations

$$\frac{\partial E_s^k}{\partial y_{j_n}^{(n),k}} = \frac{\partial\left(\sum_{i_N=1}^{m_N}\frac{1}{2}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right)^2\right)}{\partial y_{j_n}^{(n),k}} = \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot \frac{\partial y_{i_N}^{(N),k}}{\partial y_{j_n}^{(n),k}} =$$

$$= \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot F_N'\left(S_{i_N}^{(N),k}\right) \cdot \frac{\partial S_{i_N}^{(N),k}}{\partial y_{j_n}^{(n),k}} =$$

$$= \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \cdot \frac{\partial y_{i_{N-1}}^{(N-1),k}}{\partial y_{j_n}^{(n),k}} =$$

$$= \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \times$$

$$\times F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \frac{\partial S_{i_{N-1}}^{(N-1),k}}{\partial y_{j_n}^{(n),k}} = \dots = \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \times$$

$$\times F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n} =$$

$$= \varepsilon_N^k \cdot MF_N' \cdot W^{(N)} \cdot MF_{N-1}' \cdot \dots \cdot W^{(n+1)} \cdot \Delta_{j_n}^n =$$

$$= C^{(n+1)} \cdot W^{(n+1)} \cdot \Delta_{j_n}^n,$$

where

$$C^{(n)} = C^{(n+1)} \cdot W^{(n+1)} \cdot MF_n', \quad C^{(N)} = \varepsilon_N^k \cdot MF_N',$$

and $\Delta_{j_n}^n$ is zero vector-column of length n with only element equal to one in position j_n.

So the modification of "standard" values will be held accordance to formulas:

$$t_{j_n}^{(n),k} = y_{j_n}^{(n),k} - \alpha \cdot C^{(n+1)} \cdot W^{(n+1)} \cdot \Delta_{j_n}^n,\ j_n = \overline{1,m_n},$$

where training step α can be taken like a constant or a adaptive.

But for all used function the domain of outcome values is limited in to the interval $(a;b)$. So, we must observe that output values are finding in the segment $[a+\beta; b+\beta]$, where β is a little threshold. Otherwise we must take boundary values like $t_{i_n}^{(n),k}$. Mathematically that expressed in the next way:

$$t_{i_n}^{(n),k} := \begin{cases} a+\beta, & if\ t_{i_n}^{(n),k} < a+\beta \\ t_{i_n}^{(n),k}, & if\ t_{i_n}^{(n),k} \in [a+\beta, b-\beta]. \\ b-\beta, & if\ t_{i_n}^{(n),k} > a-\beta \end{cases}$$

Let's find the second order partial derivations of the error function by the output of n-th neural layer $t_{i_n}^{(n),k}$, $i_n = \overline{1,m_n}$:

$$\frac{\partial^2 E_s^k}{\partial y_{j_n}^{(n),k}\partial y_{l_n}^{(n),k}} = \frac{\partial}{\partial y_{l_n}^{(n),k}}\left(\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot F_N'\left(S_{i_N}^{(N),k}\right) \times\right.$$

$$\left.\times \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n}\right) =$$

$$= \left(\sum_{i_N=1}^{m_N}\frac{\partial\left(y_{i_N}^{(N),k} - t_{i_N}^k\right)}{\partial y_{l_n}^{(n),k}} \cdot F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \times\right.$$

$$\times F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n} +$$

$$+ \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot \frac{\partial F_N'\left(S_{i_N}^{(N),k}\right)}{\partial y_{l_n}^{(n),k}} \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \times$$

$$\times F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n} +$$

$$+ \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \times$$

$$\times \frac{\partial F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)}{\partial y_{l_n}^{(n),k}} \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n} + \dots +$$

$$+ \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \times$$

$$\left.\times F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \frac{\partial F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)}{\partial y_{l_n}^{(n),k}} \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n}\right) =$$

$$= \left(\sum_{i_N=1}^{m_N}\left(F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \times\right.\right.$$

$$\left.\times F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n}\right) \times$$

$$\times \left(F_N'\left(S_{i_N}^{(N),k}\right) \cdot \sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{l_n}^{i_n}\right) +$$

$$+ \sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k} - t_{i_N}^k\right) \cdot F_N''\left(S_{i_N}^{(N),k}\right) \times$$

$$\times \left(\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)} \cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right) \cdot \dots \cdot \sum_{i_{n+1}=1}^{m_{n+1}} w_{i_ni_{n+1}}^{(n+1)} \cdot \delta_{j_n}^{i_n}\right) \times$$

$$\times\left(\sum_{i_{N-1}=1}^{m_{N-1}} w^{(N)}_{i_{N-1}i_N}\cdot F_{N-1}'\left(S^{(N-1),k}_{i_{N-1}}\right)\cdot\ldots\cdot\sum_{i_{n+1}=1}^{m_{n+1}} w^{(n+1)}_{i_n i_{n+1}}\cdot\delta^{i_n}_{l_n}\right)+$$

$$+\sum_{i_N=1}^{m_N}\left(y^{(N),k}_{i_N}-t^{k}_{i_N}\right)\cdot F_N'\left(S^{(N),k}_{i_N}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}} w^{(N)}_{i_{N-1}i_N}\cdot F_{N-1}''\left(S^{(N-1),k}_{i_{N-1}}\right)\times$$

$$\times\left(\sum_{i_{N-2}}^{m_{N-2}} w^{(N-1)}_{i_{N-2}i_{N-1}}\ldots\cdot\sum_{i_{n+1}=1}^{m_{n+1}} w^{(n+1)}_{i_n i_{n+1}}\cdot\delta^{i_n}_{j_n}\right)\times$$

$$\times\left(\sum_{i_{N-2}}^{m_{N-2}} w^{(N-1)}_{i_{N-2}i_{N-1}}\ldots\cdot\sum_{i_{n+1}=1}^{m_{n+1}} w^{(n+1)}_{i_n i_{n+1}}\cdot\delta^{i_n}_{l_n}\right)+\ldots+$$

$$+\sum_{i_N=1}^{m_N}\left(y^{(N),k}_{i_N}-t^{k}_{i_N}\right)\cdot F_N'\left(S^{(N),k}_{i_N}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}} w^{(N)}_{i_{N-1}i_N}\cdot F_{N-1}'\left(S^{(N-1),k}_{i_{N-1}}\right)\times$$

$$\times\ldots\cdot F_{n+1}'\left(S^{(n+1),k}_{i_{n+1}}\right)\cdot\left(\sum_{i_{n+1}=1}^{m_{n+1}} w^{(n+1)}_{i_n i_{n+1}}\cdot\delta^{i_n}_{j_n}\right)\cdot\left(\sum_{i_{n+1}=1}^{m_{n+1}} w^{(n+1)}_{i_n i_{n+1}}\cdot\delta^{i_n}_{l_n}\right)\Bigg)=$$

$$=\left(W^{(n+1)}\cdot\Delta^n_{l_n}\right)^T\cdot U^{(n+1),k}\cdot\left(W^{(n+1)}\cdot\Delta^n_{j_n}\right),$$

where

$$U^{(n),k}=\left(W^{(n+1)}\cdot MF_n'\right)^T\cdot U^{(n+1),k}\cdot\left(W^{(n+1)}\cdot MF_n'\right)+W^{(n+1)}\cdot MF_n''$$

are calculated recurrently beginning from the

$$U^{(N),k}=\left(MF_N'\right)^2+DE^{(N),k}\cdot MF_n'.$$

Extending error function in to the Taylor series we receive

$$E^{(n),k}_s(t+1)=E^{(n),k}_s(t)+\sum_{j_n=1}^{m_n}\frac{\partial E^k_s}{\partial y^{(n),k}_{j_n}}\cdot\left(t^{(n),k}_{j_n}-y^{(n),k}_{j_n}\right)+$$

$$+\frac{1}{2}\cdot\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\frac{\partial^2 E^k_s}{\partial y^{(n),k}_{j_n}\partial y^{(n),k}_{l_n}}\cdot\left(t^{(n),k}_{j_n}-y^{(n),k}_{j_n}\right)\cdot\left(t^{(n),k}_{l_n}-y^{(n),k}_{l_n}\right)=$$

$$=E^{(n),k}_s(t)-\alpha\cdot\sum_{j_n=1}^{m_n}\left(\frac{\partial E^k_s}{\partial y^{(n),k}_{j_n}}\right)^2+$$

$$+\alpha^2\cdot\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\frac{\partial^2 E^k_s}{\partial y^{(n),k}_{j_n}\partial y^{(n),k}_{l_n}}\cdot\frac{\partial E^k_s}{\partial y^{(n),k}_{j_n}}\cdot\frac{\partial E^k_s}{\partial y^{(n),k}_{l_n}}=$$

$$=E^{(n),k}_s(t)-\alpha\cdot\sum_{j_n=1}^{m_n}\left(C^{(n+1)}\cdot W^{(n+1)}\cdot\Delta^n_{j_n}\right)^2+$$

$$+\frac{\alpha^2}{2}\cdot\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\left(C^{(n+1)}\cdot W^{(n+1)}\cdot\Delta^n_{j_n}\right)\times$$

$$\times\left(\left(W^{(n+1)}\cdot\Delta^n_{l_n}\right)^T\cdot U^{(n+1),k}\cdot\left(W^{(n+1)}\cdot\Delta^n_{j_n}\right)\right)\times$$

$$\times\left(C^{(n+1)}\cdot W^{(n+1)}\cdot\Delta^n_{l_n}\right)=E^{(n),k}_s(t)-\alpha\cdot\sum_{j_n=1}^{m_n}\left(C^{(n+1)}\cdot P^n_{j_n}\right)^2+$$

$$+\frac{\alpha^2}{2}\cdot\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\left(C^{(n+1)}\cdot P^n_{j_n}\right)\cdot\left(\left(P^n_{j_n}\right)^T\cdot U^{(n+1),k}\cdot\left(P^n_{l_n}\right)\right)\cdot\left(C^{(n+1)}\cdot P^n_{l_n}\right)$$

where $P^n_{j_n}=W^{(n+1)}\cdot\Delta^n_{j_n}$.

Let's find a such value of α, that minimize network error. For that purposes we must compare to zero the next expression:

$$\frac{\partial E^{(n),k}_s(t+1)}{\partial\alpha}=-\sum_{j_n=1}^{m_n}\left(C^{(n+1)}\cdot P^n_{j_n}\right)^2+$$

$$+\alpha\cdot\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\left(C^{(n+1)}\cdot P^n_{j_n}\right)\cdot\left(\left(P^n_{j_n}\right)^T\cdot U^{(n+1),k}\cdot\left(P^n_{l_n}\right)\right)\cdot\left(C^{(n+1)}\cdot P^n_{l_n}\right)$$

And finally we receive

$$\alpha=\frac{\sum_{j_n=1}^{m_n}\left(C^{(n+1)}\cdot P^n_{j_n}\right)^2}{\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\left(C^{(n+1)}\cdot P^n_{j_n}\right)\cdot\left(\left(P^n_{j_n}\right)^T\cdot U^{(n+1),k}\cdot\left(P^n_{l_n}\right)\right)\cdot\left(C^{(n+1)}\cdot P^n_{l_n}\right)}$$

Taking received values $t^{(n),k}_{i_n}$ like "standard" we can find formulas for synaptic connection modification in n-th neural layer.

Let us extend mean-square error of n-th layer in the next way:

$$E^{(n)}_S=\frac{1}{2L}\sum_{k=1}^{L}\sum_{i_n=1}^{m_n}\left(F_n\left(\sum_{i_{n-1}=1}^{m_{n-1}} w^{(n)}_{i_{n-1}i_n}y^{(n-1),k}_{i_{n-1}}-T^{(n)}_{i_n}\right)-t^{(n),k}_{i_n}\right)^2=$$

$$=\frac{1}{L}\sum_{k=1}^{L}E^{(n),k}_s.$$

Then

$$\frac{\partial E^{(n),k}_s}{\partial w^{(n)}_{j_{n-1}j_n}}=\frac{\partial\left(\sum_{i_n=1}^{m_n}\frac{1}{2}\left(y^{(n),k}_{i_n}-t^{(n),k}_{i_n}\right)^2\right)}{\partial w^{(n)}_{j_{n-1}j_n}}=$$

$$=\sum_{i_n=1}^{m_n}\left(y^{(n),k}_{i_n}-t^{(n),k}_{i_n}\right)\cdot\frac{\partial y^{(n),k}_{i_n}}{\partial w^{(n)}_{j_{n-1}j_n}}=$$

$$=\sum_{i_n=1}^{m_n}\left(y^{(n),k}_{i_n}-t^{(n),k}_{i_n}\right)\cdot F_n'\left(S^{(n),k}_{i_n}\right)\cdot\frac{\partial S^{(n),k}_{i_n}}{\partial w^{(n)}_{j_{n-1}j_n}}=$$

$$=\sum_{i_n=1}^{m_n}\left(y^{(n),k}_{i_n}-t^{(n),k}_{i_n}\right)\cdot F_n'\left(S^{(n),k}_{i_n}\right)\cdot y^{(n-1),k}_{j_{n-1}}\cdot\delta^{i_n}_{j_n},$$

where

$$\delta^{i_n}_{j_n}=\begin{cases}1, & i_n=j_n\\ 0, & i_n\neq j_n\end{cases}.$$

Using matrix algoritnmization we can rewrite above formulas in the next way:

$$\frac{\partial E^{(n),k}_s}{\partial w^{(n)}_{j_{n-1}j_n}}=\sum_{i_n=1}^{m_n}\left(y^{(n),k}_{i_n}-t^{(n),k}_{i_n}\right)\cdot F_n'\left(S^{(n),k}_{i_n}\right)\cdot y^{(n-1),k}_{j_{n-1}}\cdot\delta^{i_n}_{j_n}=$$

$$=\varepsilon^k_n\cdot MF_n'\cdot M^{(n)}_{j_nj_{n-1}}\cdot Y^{(n-1),k}=C^{(n),k}_{layer}\cdot K^{(n),k}_{j_{n-1}j_n}.$$

In a similar manner

$$\frac{\partial E^{(n),k}_s}{\partial T^{(n)}_{j_n}}=\frac{\partial\left(\sum_{i_n=1}^{m_n}\frac{1}{2}\left(y^{(n),k}_{i_n}-t^{(n),k}_{i_n}\right)^2\right)}{\partial T^{(n)}_{j_n}}=$$

$$= \sum_{i_n=1}^{m_n} \left(y_{i_n}^{(n),k} - t_{i_n}^{(n),k} \right) \cdot \frac{\partial y_{i_n}^{(n),k}}{\partial T_{j_n}^{(n)}} =$$

$$= \sum_{i_n=1}^{m_n} \left(y_{i_n}^{(n),k} - t_{i_n}^{(n),k} \right) \cdot F_n'\left(S_{i_n}^{(n),k} \right) \cdot \frac{\partial S_{i_n}^{(n),k}}{\partial T_{j_n}^{(n)}} =$$

$$= \sum_{i_n=1}^{m_n} \left(y_{i_n}^{(n),k} - t_{i_n}^{(n),k} \right) \cdot F_n'\left(S_{i_n}^{(n),k} \right) \cdot (-1) \cdot \delta_{j_n}^{i_n} =$$

$$= \varepsilon_n^k \cdot MF_n' \cdot M_{j_n(m_{n-1}+1)}^{(n)} \cdot Y^{(n-1),k} = C_{layer}^{(n),k} \cdot K_{(m_{n-1}+1)\,j_n}^{(n),k},$$

where $C_{layer}^{(n),k} = \varepsilon_n^k \cdot MF_n'$, $K_{ij}^{(n),k} = M_{ji}^{(n)} \cdot Y^{(n-1),k}$.

So, the formulas for weights and threshold changes will be the next:

$$w_{j_{n-1}j_n}^{(n)}(t+1) = w_{j_{n-1}j_n}^{(n)}(t) - \alpha \cdot \frac{1}{L} \cdot \sum_{k=1}^{L} \varepsilon_n^k \cdot MF_n' \cdot M_{j_n j_{n-1}}^{(n)} \cdot Y^{(n-1),k}$$

$$T_{j_n}^{(n)}(t+1) = T_{j_n}^{(n)}(t) - \alpha \cdot \frac{1}{L} \cdot \sum_{k=1}^{L} \varepsilon_n^k \cdot MF_n' \cdot M_{j_n(m_{n-1}+1)}^{(n)} \cdot Y^{(n-1),k}$$

for $j_{n-1} = \overline{1, m_{n-1}}$, $j_n = \overline{1, m_n}$, or

$$w_{j_{n-1}j_n}^{(n)}(t+1) = w_{j_{n-1}j_n}^{(n)}(t) - \alpha \cdot \frac{1}{L} \cdot G_{j_{n-1}j_n}^{(n),layer},$$

$$T_{j_n}^{(n)}(t+1) = T_{j_n}^{(n)}(t) - \alpha \cdot \frac{1}{L} \cdot G_{j_{n-1}j_n}^{(n),layer},$$

where $G_{j_{n-1}j_n}^{(n),layer} = \sum_{k=1}^{L} C_{layer}^{(n),k} \cdot K_{j_{n-1}j_n}^{(n),k}$.

Let's find second order partial derivatives of error function:

$$\frac{\partial^2 E_s^{(n),k}}{\partial w_{j_{n-1}j_n}^{(n)} \partial w_{l_{n-1}l_n}^{(n)}} =$$

$$= \frac{\partial}{\partial w_{l_{n-1}l_n}} \left(\sum_{i_n=1}^{m_n} \left(y_{i_n}^{(n),k} - t_{i_n}^{(n),k} \right) \cdot F_n'\left(S_{i_n}^{(n),k} \right) \cdot y_{j_{n-1}}^{(n-1),k} \cdot \delta_{j_n}^{i_n} \right) =$$

$$= \sum_{i_n=1}^{m_n} \left(\frac{\partial\left(\left(y_{i_n}^{(n),k} - t_{i_n}^{(n),k} \right) \right)}{\partial w_{l_{n-1}l_n}} \cdot F_n'\left(S_{i_n}^{(n),k} \right) \cdot y_{j_{n-1}}^{(n-1),k} \cdot \delta_{j_n}^{i_n} + \right.$$

$$\left. + \left(y_{i_n}^{(n),k} - t_{i_n}^{(n),k} \right) \cdot \frac{\partial\left(F_n'\left(S_{i_n}^{(n),k} \right) \right)}{\partial w_{l_{n-1}l_n}} \cdot y_{j_{n-1}}^{(n-1),k} \cdot \delta_{j_n}^{i_n} \right) =$$

$$= \sum_{i_n=1}^{m_n} \left(\left(F_n'\left(S_{i_n}^{(n),k} \right) \right)^2 \cdot \left(y_{l_{n-1}}^{(n-1),k} \cdot \delta_{l_n}^{i_n} \right) \cdot \left(y_{j_{n-1}}^{(n-1),k} \cdot \delta_{j_n}^{i_n} \right) + \right.$$

$$\left. + \left(y_{i_n}^{(n),k} - t_{i_n}^{(n),k} \right) \cdot F_n''\left(S_{i_n}^{(n),k} \right) \cdot \left(y_{l_{n-1}}^{(n-1),k} \cdot \delta_{l_n}^{i_n} \right) \cdot \left(y_{j_{n-1}}^{(n-1),k} \cdot \delta_{j_n}^{i_n} \right) \right) =$$

$$= \left(M_{l_n l_{n-1}}^{(n)} \cdot Y^{(n-1),k} \right)^T \cdot \left(\left(MF_n' \right)^2 + DE^{(n),k} \cdot MF_n'' \right) \times$$

$$\times \left(M_{j_n j_{n-1}}^{(n)} \cdot Y^{(n-1),k} \right) =$$

$$= \left(K_{l_{n-1}l_n}^{(n),k} \right)^T \cdot \left(\left(MF_n' \right)^2 + DE^{(n),k} \cdot MF_n'' \right) \cdot K_{j_{n-1}j_n}^{(n),k}.$$

In the same manner we receive

$$\frac{\partial^2 E_s^{(n),k}}{\partial T_{j_n}^{(n)} \partial T_{l_n}^{(n)}} =$$

$$= \left(K_{(m_{n-1}+1)l_n}^{(n),k} \right)^T \cdot \left(\left(MF_n' \right)^2 + DE^{(n),k} \cdot MF_n'' \right) \cdot K_{(m_{n-1}+1)\,j_n}^{(n),k}$$

$$\frac{\partial^2 E_s^{(n),k}}{\partial w_{j_{n-1}j_n}^{(n)} \partial T_{l_n}^{(n)}} =$$

$$= \left(K_{(m_{n-1}+1)l_n}^{(n),k} \right)^T \cdot \left(\left(MF_n' \right)^2 + DE^{(n),k} \cdot MF_n'' \right) \cdot K_{j_{n-1}j_n}^{(n),k}.$$

After modification of n-th layer synaptic connections the network error is changed accordance to the formulas:

$$E_S^{(n)}(t+1) = \frac{1}{L} \cdot \sum_{k=1}^{L} E_s^{(n),k}(t+1) = \frac{1}{L} \sum_{k=1}^{L} E_s^{(n),k}(t) +$$

$$+ \frac{1}{L} \cdot \left(\sum_{j_{n-1}=1}^{m_{n-1}} \sum_{j_n=1}^{m_n} \left(\sum_{k=1}^{L} \frac{\partial E_s^{(n),k}}{\partial w_{j_{n-1}j_n}^{(n)}} \right) \cdot \left(w_{j_{n-1}j_n}^{(n)}(t+1) - w_{j_{n-1}j_n}^{(n)}(t) \right) + \right.$$

$$\left. + \sum_{j_n=1}^{m_n} \left(\sum_{k=1}^{L} \frac{\partial E_s^{(n),k}}{\partial T_{j_n}^{(n)}} \right) \cdot \left(T_{j_n}^{(n)}(t+1) - T_{j_n}^{(n)}(t) \right) \right) =$$

$$+ \frac{1}{2L} \cdot \sum_{k=1}^{L} \left(\sum_{j_{n-1}=1}^{m_{n-1}} \sum_{j_n=1}^{m_n} \sum_{l_{n-1}=1}^{m_{n-1}} \sum_{l_n=1}^{m_n} \frac{\partial^2 E_s^{(n),k}}{\partial w_{j_{n-1}j_n}^{(n)} \partial w_{l_{n-1}l_n}^{(n)}} \times \right.$$

$$\times \left(w_{j_{n-1}j_n}^{(n)}(t+1) - w_{j_{n-1}j_n}^{(n)}(t) \right) \cdot \left(w_{l_{n-1}l_n}^{(n)}(t+1) - w_{l_{n-1}l_n}^{(n)}(t) \right) +$$

$$+ \sum_{j_{n-1}=1}^{m_{n-1}} \sum_{j_n=1}^{m_n} \sum_{l_n=1}^{m_n} \frac{\partial^2 E_s^{(n),k}}{\partial w_{j_{n-1}j_n}^{(n)} \partial T_{l_n}^{(n)}} \cdot \left(w_{j_{n-1}j_n}^{(n)}(t+1) - w_{j_{n-1}j_n}^{(n)}(t) \right) \times$$

$$\times \left(T_{l_n}^{(n)}(t+1) - T_{l_n}^{(n)}(t) \right) +$$

$$+ \sum_{j_n=1}^{m_n} \sum_{l_{n-1}=1}^{m_{n-1}} \sum_{l_n=1}^{m_n} \frac{\partial^2 E_s^{(n),k}}{\partial T_{j_n}^{(n)} \partial w_{l_{n-1}l_n}^{(n)}} \cdot \left(T_{j_n}^{(n)}(t+1) - T_{j_n}^{(n)}(t) \right) \times$$

$$\times \left(w_{l_{n-1}l_n}^{(n)}(t+1) - w_{l_{n-1}l_n}^{(n)}(t) \right) +$$

$$+ \sum_{j_n=1}^{m_n} \sum_{l_n=1}^{m_n} \frac{\partial^2 E_s^{(n),k}}{\partial T_{j_n}^{(n)} \partial T_{l_n}^{(n)}} \cdot \left(T_{j_n}^{(n)}(t+1) - T_{j_n}^{(n)}(t) \right) \times$$

$$\left. \times \left(T_{l_n}^{(n)}(t+1) - T_{l_n}^{(n)}(t) \right) \right) =$$

$$= E_S^{(n)}(t) - \alpha \cdot \frac{1}{L^2} \cdot \sum_{j_{n-1}=1}^{m_{n-1}+1} \sum_{j_n=1}^{m_n} \left(G_{j_{n-1}j_n}^{(n),layer} \right)^2 +$$

$$+ \alpha^2 \cdot \frac{1}{2L^3} \cdot \left(\sum_{j_{n-1}=1}^{m_{n-1}+1} \sum_{j_n=1}^{m_n} \sum_{l_{n-1}=1}^{m_{n-1}+1} \sum_{l_n=1}^{m_n} \left(G_{l_{n-1}l_n}^{(n),layer} \times \right. \right.$$

$$\left. \left. \times \sum_{k=1}^{L} \left(\left(K_{l_{n-1}l_n}^{(n),k} \right)^T \cdot \left(\left(MF_n' \right)^2 + DE^{(n),k} \cdot MF_n'' \right) \cdot K_{j_{n-1}j_n}^{(n),k} \right) \cdot G_{j_{n-1}j_n}^{(n),layer} \right) \right)$$

Let's find such point α, in which error function reach it minimal value. For that purpose we mast compare with zero the next expression

$$\frac{\partial E_S^{(n)}(t+1)}{\partial \alpha} = \frac{1}{L^2} \cdot \sum_{j_{n-1}=1}^{m_{n-1}+1} \sum_{j_n=1}^{m_n} \left(G_{j_{n-1}j_n}^{(n),layer} \right)^2 +$$

$$+\alpha\cdot\frac{1}{L^3}\cdot\left(\sum_{j_{n-1}=1}^{m_{n-1}+1}\sum_{j_n=1}^{m_n}\sum_{l_{n-1}=1}^{m_{n-1}+1}\sum_{l_n=1}^{m_n}\left(G_{l_{n-1}l_n}^{(n),layer}\times\right.\right.$$

$$\left.\left.\times\sum_{k=1}^{L}\left(\left(K_{l_{n-1}l_n}^{(n),k}\right)^T\cdot\left(\left(MF_n'\right)^2+DE^{(n),k}\cdot MF_n''\right)\cdot K_{j_{n-1}j_n}^{(n),k}\right)\cdot G_{j_{n-1}j_n}^{(n),layer}\right)\right)$$

In that way we receive the next value

$$\alpha=\frac{L\cdot\sum_{j_{n-1}=1}^{m_{n-1}+1}\sum_{j_n=1}^{m_n}\left(G_{j_{n-1}j_n}^{(n),layer}\right)^2}{\sum_{j_{n-1},l_{n-1}=1}^{m_{n-1}+1}\sum_{j_n,l_n=1}^{m_n}G_{l_{n-1}l_n}^{(n),layer}\cdot\left(S_{(n)}\right)_{l_{n-1}l_l}^{j_{n-1}j_n}\cdot G_{l_{n-1}l_n}^{(n),layer}},$$

where

$$\left(S_{(n)}\right)_{l_{n-1}l_l}^{j_{n-1}j_n}=\sum_{k=1}^{L}\left(\left(K_{l_{n-1}l_n}^{(n),k}\right)^T\cdot\left(\left(MF_n'\right)^2+DE^{(n),k}\cdot MF_n''\right)\cdot K_{j_{n-1}j_n}^{(n),k}\right)$$

And finally

$$\alpha^{(n)}=\alpha\cdot\frac{1}{L}=\frac{\sum_{j_{n-1}=1}^{m_{n-1}+1}\sum_{j_n=1}^{m_n}\left(G_{j_{n-1}j_n}^{(n),layer}\right)^2}{\sum_{j_{n-1},l_{n-1}=1}^{m_{n-1}+1}\sum_{j_n,l_n=1}^{m_n}G_{l_{n-1}l_n}^{(n),layer}\cdot\left(S_{(n)}\right)_{l_{n-1}l_l}^{j_{n-1}j_n}\cdot G_{l_{n-1}l_n}^{(n),layer}}$$

■

4. LAYERWISE TRAINING OF THE MULTILAYER NEURAL NETWORK WITH USE OF THE ADAPTIVE TRAINING STEP

Let us represent layerwise training technique, which is an extension from such method for twolayer network [2]. The algorithm of thus method can be represented in the next way:

Procedure Network training
begin
set training accuracy ε
repeat
for n=N **down to** 1 **do**
begin
finding the error E_S for all training set;
modification of n-th layer synaptic connection
end
until $E_S<\varepsilon$
end.

For the faster convergence of this algorithm we can take a training step like adaptive. It calculation is based on the next theorem.

Theorem. For the layerwise training methodic of multilayer heterogeneous feedforward neural network the adaptive training steps for each layer are calculated accordance to the formulas

$$\alpha^{(n)}=\frac{L\cdot\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\left(\sum_{k=1}^{L}C^{(n)}\cdot M_{j_nj_{n-1}}^{(n)}\cdot Y^{(n-1),k}\right)}{\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\sum_{l_{n-1}=1}^{m_{n-1}}\sum_{l_n=1}^{m_n}\left(\sum_{k=1}^{L}\left(\left(K_{l_{n-1}l_n}^{(n),k}\right)^T\cdot U^{(n),k}\cdot\left(K_{j_{n-1}j_n}^{(n),k}\right)\right)\right)}$$

where

$$K_{j_{n-1}j_n}^{(n),k}=M_{j_nj_{n-1}}^{(n)}\cdot Y^{(n-1),k}$$

and

$$U^{(n),k}=\left(W^{(n+1)}\cdot MF_n'\right)^T\cdot U^{(n+1),k}\cdot\left(W^{(n+1)}\cdot MF_n'\right)+W^{(n+1)}\cdot MF_n''$$

are computed recurrently from the

$$U^{(N),k}=\left(MF_N'\right)^2+DE^{(N),k}\cdot MF_n'.$$

Proof. Accordance to [1] we have:

$$\frac{\partial E_s^{(k)}}{\partial w_{j_{n-1}j_n}^{(n)}}=\frac{\partial\left(\sum_{i_N=1}^{m_N}\frac{1}{2}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)^2\right)}{\partial w_{j_{n-1}j_n}^{(n)}}=$$

$$=\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}}w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots\cdot$$

$$\cdot\ldots\cdot F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)\cdot\sum_{i_n=1}^{m_n}w_{i_ni_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}=$$

$$=C^{(n)}\cdot M_{j_nj_{n-1}}^{(n)}\cdot Y^{(n-1),k},$$

and by the analogy

$$\frac{\partial E_s^{(k)}}{\partial T_{j_n}^{(n)}}=C^{(n)}\cdot M_{j_n(m_{n-1}+1)}^{(n)}\cdot Y^{(n-1),k}.$$

Let's find second order partial derivation of error function:

$$\frac{\partial^2E_s^{(k)}}{\partial w_{j_{n-1}j_n}^{(n)}\partial w_{l_{n-1}l_n}^{(n)}}=\frac{\partial}{\partial w_{l_{n-1}l_n}^{(n)}}\left(\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot F_N'\left(S_{i_N}^{(N),k}\right)\times\right.$$

$$\times\sum_{i_{N-1}=1}^{m_{N-1}}w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots\cdot$$

$$\left.\cdot\ldots\cdot F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)\cdot\sum_{i_n=1}^{m_n}w_{i_ni_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}\right)=$$

$$=\left(\sum_{i_N=1}^{m_N}\frac{\partial\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)}{\partial w_{l_{n-1}l_n}^{(n)}}\cdot F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}}w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots\cdot\right.$$

$$\cdot\ldots\cdot F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)\cdot\sum_{i_n=1}^{m_n}w_{i_ni_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}+$$

$$+\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot\frac{\partial F_N'\left(S_{i_N}^{(N),k}\right)}{\partial w_{l_{n-1}l_n}^{(n)}}\cdot\sum_{i_{N-1}=1}^{m_{N-1}}w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots\cdot$$

$$\cdot\ldots\cdot F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)\cdot\sum_{i_n=1}^{m_n}w_{i_ni_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}+$$

$$+\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}}w_{i_{N-1}i_N}^{(N)}\cdot\frac{\partial F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)}{\partial w_{l_{n-1}l_n}^{(n)}}\cdot\ldots\cdot$$

$$\cdot\ldots\cdot F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)\cdot\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}+$$

$$+\ldots+$$

$$+\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots$$

$$\left.\cdot\ldots\cdot F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)\cdot\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot\frac{\partial F_n'\left(S_{i_n}^{(n),k}\right)}{\partial w_{l_{n-1}l_n}^{(n)}}\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}\right)=$$

$$=\left(\sum_{i_N=1}^{m_N}\left(F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\times\right.\right.$$

$$\left.\times\ldots\cdot\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}\right)\times$$

$$\times\left(F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\times\right.$$

$$\left.\times\ldots\cdot\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{l_{n-1}}^{(n-1),k}\cdot\delta_{l_n}^{i_n}\right)+$$

$$+\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot F_N''\left(S_{i_N}^{(N),k}\right)\times$$

$$\times\left(\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots\times\right.$$

$$\left.\times\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}\right)\times$$

$$\times\left(\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots\times\right.$$

$$\left.\times\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{l_{n-1}}^{(n-1),k}\cdot\delta_{l_n}^{i_n}\right)+$$

$$+\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}''\left(S_{i_{N-1}}^{(N-1),k}\right)\times$$

$$\times\left(\ldots\cdot\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}\right)\times$$

$$\times\left(\ldots\cdot\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n'\left(S_{i_n}^{(n),k}\right)\cdot y_{l_{n-1}}^{(n-1),k}\cdot\delta_{l_n}^{i_n}\right)+\ldots+$$

$$+\sum_{i_N=1}^{m_N}\left(y_{i_N}^{(N),k}-t_{i_N}^k\right)\cdot F_N'\left(S_{i_N}^{(N),k}\right)\cdot\sum_{i_{N-1}=1}^{m_{N-1}} w_{i_{N-1}i_N}^{(N)}\cdot F_{N-1}'\left(S_{i_{N-1}}^{(N-1),k}\right)\cdot\ldots\times$$

$$\times\ldots\cdot F_{n+1}'\left(S_{i_{n+1}}^{(n+1),k}\right)\cdot\sum_{i_n=1}^{m_n} w_{i_n i_{n+1}}^{(n+1)}\cdot F_n''\left(S_{i_n}^{(n),k}\right)\times$$

$$\left.\times\left(y_{j_{n-1}}^{(n-1),k}\cdot\delta_{j_n}^{i_n}\right)\cdot\left(y_{l_{n-1}}^{(n-1),k}\cdot\delta_{l_n}^{i_n}\right)\right)=$$

$$=\left(M_{l_n l_{n-1}}^{(n)}\cdot Y^{(n-1),k}\right)^T\cdot U^{(n),k}\cdot\left(M_{j_n j_{n-1}}^{(n)}\cdot Y^{(n-1),k}\right)=$$

$$=\left(K_{l_{n-1}l_n}^{(n),k}\right)^T\cdot U^{(n),k}\cdot\left(K_{j_{n-1}j_n}^{(n),k}\right),$$

where

$$K_{j_{n-1}j_n}^{(n),k}=M_{j_n j_{n-1}}^{(n)}\cdot Y^{(n-1),k}$$

and

$$U^{(n),k}=\left(W^{(n+1)}\cdot MF_n'\right)^T\cdot U^{(n+1),k}\cdot\left(W^{(n+1)}\cdot MF_n'\right)+W^{(n+1)}\cdot MF_n''$$

are computed recurrently from the

$$U^{(N),k}=\left(MF_N'\right)^2+DE^{(N),k}\cdot MF_n'.$$

By the same manner we receive:

$$\frac{\partial^2 E_s^{(k)}}{\partial w_{j_{n-1}j_n}^{(n)}\partial T_{l_n}^{(n)}}=\left(K_{(m_{n-1}+1)l_n}^{(n),k}\right)^T\cdot U^{(n),k}\cdot\left(K_{j_{n-1}j_n}^{(n),k}\right),$$

$$\frac{\partial^2 E_s^{(k)}}{\partial T_{j_n}^{(n)}\partial T_{l_n}^{(n)}}=\left(K_{(m_{n-1}+1)l_n}^{(n),k}\right)^T\cdot U^{(n),k}\cdot\left(K_{(m_{n-1}+1)j_n}^{(n),k}\right).$$

Let's extend error function in to the Taylor series:

$$E_S(t+1)=\frac{1}{L}\cdot\sum_{k=1}^{L}E_s^k(t+1)=\frac{1}{L}\sum_{k=1}^{L}E_s^k(t)+$$

$$+\frac{1}{L}\cdot\left(\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\left(\sum_{k=1}^{L}\frac{\partial E_s^k}{\partial w_{j_{n-1}j_n}^{(n)}}\right)\cdot\left(w_{j_{n-1}j_n}^{(n)}(t+1)-w_{j_{n-1}j_n}^{(n)}(t)\right)+\right.$$

$$\left.+\sum_{j_n=1}^{m_n}\left(\sum_{k=1}^{L}\frac{\partial E_s^k}{\partial T_{j_n}^{(n)}}\right)\cdot\left(T_{j_n}^{(n)}(t+1)-T_{j_n}^{(n)}(t)\right)\right)=$$

$$+\frac{1}{2L}\cdot\sum_{k=1}^{L}\left(\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\sum_{l_{n-1}=1}^{m_{n-1}}\sum_{l_n=1}^{m_n}\frac{\partial^2 E_s^k}{\partial w_{j_{n-1}j_n}^{(n)}\partial w_{l_{n-1}l_n}^{(n)}}\times\right.$$

$$\times\left(w_{j_{n-1}j_n}^{(n)}(t+1)-w_{j_{n-1}j_n}^{(n)}(t)\right)\cdot\left(w_{l_{n-1}l_n}^{(n)}(t+1)-w_{l_{n-1}l_n}^{(n)}(t)\right)+$$

$$+\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\frac{\partial^2 E_s^k}{\partial w_{j_{n-1}j_n}^{(n)}\partial T_{l_n}^{(n)}}\cdot\left(w_{j_{n-1}j_n}^{(n)}(t+1)-w_{j_{n-1}j_n}^{(n)}(t)\right)\times$$

$$\times\left(T_{l_n}^{(n)}(t+1)-T_{l_n}^{(n)}(t)\right)+$$

$$+\sum_{j_n=1}^{m_n}\sum_{l_{n-1}=1}^{m_{n-1}}\sum_{l_n=1}^{m_n}\frac{\partial^2 E_s^k}{\partial T_{j_n}^{(n)}\partial w_{l_{n-1}l_n}^{(n)}}\cdot\left(T_{j_n}^{(n)}(t+1)-T_{j_n}^{(n)}(t)\right)\times$$

$$\times\left(w_{l_{n-1}l_n}^{(n)}(t+1)-w_{l_{n-1}l_n}^{(n)}(t)\right)+$$

$$+\sum_{j_n=1}^{m_n}\sum_{l_n=1}^{m_n}\frac{\partial^2 E_s^k}{\partial T_{j_n}^{(n)}\partial T_{l_n}^{(n)}}\cdot\left(T_{j_n}^{(n)}(t+1)-T_{j_n}^{(n)}(t)\right)\times$$

$$\left.\times\left(T_{l_n}^{(n)}(t+1)-T_{l_n}^{(n)}(t)\right)\right)=$$

$$=E_S(t)-\alpha^{(n)}\cdot\frac{1}{L^2}\cdot\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\left(\sum_{k=1}^{L}C^{(n)}\cdot M_{j_n j_{n-1}}^{(n)}\cdot Y^{(n-1),k}\right)+$$

$$+\left(\alpha^{(n)}\right)^2\cdot\frac{1}{2L^3}\cdot\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\sum_{l_{n-1}=1}^{m_{n-1}}\sum_{l_n=1}^{m_n}\left(\sum_{k=1}^{L}\left(\left(K_{l_{n-1}l_n}^{(n),k}\right)^T\cdot U^{(n),k}\cdot\left(K_{j_{n-1}j_n}^{(n),k}\right)\right)\right)$$

For finding minima of error function we take a first derivation by $\alpha^{(n)}$ and equal it to zero. So, we receive:

$$\alpha^{(n)}=\frac{L\cdot\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\left(\sum_{k=1}^{L}C^{(n)}\cdot M_{j_n j_{n-1}}^{(n)}\cdot Y^{(n-1),k}\right)}{\sum_{j_{n-1}=1}^{m_{n-1}}\sum_{j_n=1}^{m_n}\sum_{l_{n-1}=1}^{m_{n-1}}\sum_{l_n=1}^{m_n}\left(\sum_{k=1}^{L}\left(\left(K_{l_{n-1}l_n}^{(n),k}\right)^T\cdot U^{(n),k}\cdot\left(K_{j_{n-1}j_n}^{(n),k}\right)\right)\right)}$$

5. EXPERIMENTS AND DISCUSSION

The results of use of the above training methods for training of twolayer feedforward neural networks with architecture 3-4-1 with sigmoid element in a hidden layer for Henon attractor forecasting are presented below. After 100 iteration was received the next middle square errors: for constant training steps $\alpha_1 = 0.02$, $\alpha_2 = 0.02$ - 0.3433; for constant training step $\alpha_1 = 0.01$, $\alpha_2 = 0.02$ - 0.3127; for constant training step $\alpha_1 = 0.02$, $\alpha_2 = 0.05$ - 0.3165; for a adaptive training step, used in Matlab Neural Network Toolbox – 0.5094; for training with Rprop – 0.09722; for training algorithm based on the networks' error conditional optimization – 0.2715; for layerwise training with use of the adaptive training step – 0.0025. For some constant training step was observed divergence of the training process. Above results shows that proposed in this paper algorithms give good convergence in the set of gradient descent methods. The same distinction of MSE was also obtained for other training sets.

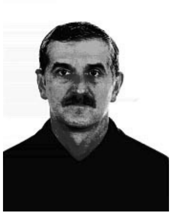
6. CONCLUSION

Implementation of such training methodics for neural network training gives a good result in a time of convergence. The matrix algorithmization of the training process is very helpful in its program realization.

7. REFERENCES

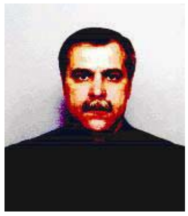
[1] N. Maniakov, L. Makhnist. Matrix algorithmization of multilayer neural networks' training process with use of gradient descents methods, *Vestnik BGTU* 5 (17) (2002). p. 60-64

[2] V. Golovko, N. Maniakov, L. Makhnist. Multilayer Neural Networks Training Methodic. *Proceedings of the Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2003)*, Lviv, Ukraine, 8-10 September 2003, pp. 185-190



Leonid Makhnist *completed studies in the mechanical-mathematical faculty of Belarusian State University in 1982. From 1992 he worked at the Brest branch of Institute of Technical Cybernetic. Since 1994 he worked in Brest State Technical University. In 1999 he received PhD degree. From 2000 Leonid Makhnist worked as Assistant Professor at the Department of High Mathematics of BSTU. He has published over 60 journal and conference paper. His current research area includes neural networks training methodics, optimization theory, programming methodology.*

Nikolaj Maniakov *graduated mechanical-mathematical faculty of Belarusian State University in 1998. At present he worked as senior lecturer at the High Mathematical Department of Brest State Technical University and makes his PhD research. He is co-author of about 30 papers. His interests are in the field of neural networks, fuzzy logic, evolutional computation and wavelets theory.*



Dr. Vladimir Rubanov *graduated mechanical-mathematical faculty of Belarusian State University in 1977. From 1977 he worked as a lecturer at Brest State Technical University. Since 1997 he is a head of High Mathematics department. He has published over 50 scientific papers. His research interesting includes differential geometry, nonlinear dynamics and chaotic process.*