ISSN 1561-2430 (Print)
ISSN 2524-2415 (Online)
UDC 519.237
https://doi.org/10.29235/1561-2430-2026-62-2-149-163

Received 16.03.2026
Поступила в редакцию 16.03.2026

Vladimir S. Mukha

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

ALGORITHM OF THE MULTILAYER ARTIFICIAL FEED-FORWARD NEURAL NETWORK LEARNING

Abstract. The algorithm of the multilayer feed-forward neural network learning is developed. The algorithm realizes an error back-propagation idea. The particularity of the algorithm is its multidimensional-matrix form, which provides its theoretical and algorithmic generality. The program realization of the algorithm is performed as the function of Matlab programming language. In spite of the multidimensional-matrix form of the algorithm, this function is defined fully by the usual matrices. The validity of the algorithm is confirmed by the computer simulation of different approximation problems including the problem of pattern recognition.

Keywords: feed-forward neural network, error back-propagation, deep learning

For citation. Mukha V. S. Algorithm of the multilayer artificial feed-forward neural network learning. *Vestsi Natsyyanal'nai akademii navuk Belarusi. Seryya fizika-matematychnykh navuk = Proceedings of the National Academy of Sciences of Belarus. Physics and Mathematics series*, 2026, vol. 62, no. 2, pp. 149–163. https://doi.org/10.29235/1561-2430-2026-62-2-149-163

В. С. Муха

Белорусский государственный университет информатики и радиоэлектроники, Минск, Республика Беларусь

АЛГОРИТМ ОБУЧЕНИЯ МНОГОСЛОЙНОЙ НЕЙРОННОЙ СЕТИ ПРЯМОГО РАСПРОСТРАНЕНИЯ

Аннотация. Разработан алгоритм обучения многослойной нейронной сети прямого распространения (feed-forward neural network), который реализует идею так называемого метода обратного распространения ошибок (error back-propagation). Особенностью алгоритма является его многомерно-матричная форма, обеспечивающая ему теоретическую и алгоритмическую общность. Выполнена программная реализация алгоритма как функции языка программирования Matlab. Несмотря на многомерно-матричную форму алгоритма, эта функция полностью определяется обычными матрицами. Корректность алгоритма подтверждена компьютерным моделированием на примере различных задач аппроксимации, включая задачу распознавания образов.

Ключевые слова: нейронная сеть прямого распространения, обратное распространение ошибки, глубокое обучение

Для цитирования. Муха, В. С. Алгоритм обучения многослойной нейронной сети прямого распространения / В. С. Муха // Весці Нацыянальнай акадэміі навук Беларусі. Серыя фізіка-матэматычных навук. – 2026. – Т. 62, № 2. – С. 149–163. https://doi.org/10.29235/1561-2430-2026-62-2-149-163

Introduction. Nowadays, neural networks increasingly being used to solve various problems instead of traditional mathematical methods. A certain euphoria is observed in the popular literature especially in the student environment about the advantages of artificial neural networks compared with the traditional methods (see, for example, [1]). It is explained by the apparent simplicity of the use of artificial neural network: let us choose the features, collect the training set and the artificial neural network will achieve everything else. On the other hand, there are some works in which a real comparative analysis of classical methods and artificial neural networks is performed for solving the specific problems [2–5]. The main difficulty on this path is the development of learning algorithm of the artificial neural network. Therefore, ready-made software products which contain learning algorithms of the artificial neural networks are usually used. However, it is advisable to have clear and easy for programming algorithms of the artificial neural networks learning for the expanding of arsenal of typical solutions in the field of artificial neural networks.

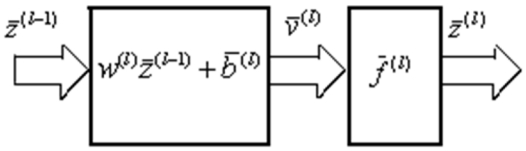

Fig. 1. The separate layer of the artificial neural network

The most popular method for the artificial neural networks learning is so called error back-propagation method [6; 7]. There are numerous descriptions of this method, but all of them are not brought to a strict algorithm suitable for error-free programming. In this paper, the solution of this problem is given. The developing algorithm bases on the idea of error back-propagation method but this terminology is not used since the algorithm does not contain any error propagation in the literal sense. By author's understanding, it is the gradient method for finding the minimum of the loss function with a specific but natural way to calculate the gradient for this task.

It was required to involve the multidimensional-matrix mathematical approach for developing the algorithm.

1. Theoretical part.

The separate layer of the artificial neural network has the form represented in Fig. 1.

In Figure 1, $\bar{z}^{(l-1)}=\left(z_1^{(l-1)},z_2^{(l-1)},...,z_{s_{l-1}}^{(l-1)}\right)$ is the input vector of the l-th layer, $\bar{v}^{(l)}=\left(v_1^{(l)},v_2^{(l)},...,v_{s_l}^{(l)}\right)$ is the output vector of the affine transformation of the l-th layer, $\bar{z}^{(l)}=\left(z_1^{(l)},z_2^{(l)},...,z_{s_l}^{(l)}\right)$ is the output vector of the activation function $\bar{f}^{(l)}$ of the l-th layer, $w^{(l)}=\left(w_{i,j}^{(l)}\right)$, $i=1,2,...,s_l$, $j=1,2,...,s_{l-1}$, is $(s_l \times s_{l-1})$-matrix of the weigth coefficients of the linear transformatin of the l-th layer, $\bar{b}^{(l)}=\left(b_1^{(l)},b_2^{(l)},...,b_{s_l}^{(l)}\right)$ is the bias vector of the l-th layer.

Such a layer is described mathematically by the following expression:

$$\bar{z}^{(l)}=\bar{f}^{(l)}(\bar{v}^{(l)}), \tag{1}$$

where

$$\bar{v}^{(l)}=w^{(l)}\bar{z}^{(l-1)}+\bar{b}^{(l)}, \tag{2}$$

$\bar{f}^{(l)}=\left(f_1^{(l)},f_2^{(l)},...,f_{s_l}^{(l)}\right)$ is the vector function which maps the vector $\bar{v}^{(l)}=\left(v_1^{(l)},v_2^{(l)},...,v_{s_l}^{(l)}\right)$ to the vector $\bar{z}^{(l)}=\left(z_1^{(l)},z_2^{(l)},...,z_{s_l}^{(l)}\right)$. This function represents the set of independent scalar functions of the scalar variables, i. e. $z_i^{(l)}=f_i^{(l)}\left(v_i^{(l)}\right)$, $i=1,2,...,s_l$. We will call two numbers (s_{l-1},s_l) the size of the l-th layer.

Let us note that we will use the multidimensional-matrix notation [8], so the product $w^{(l)}\bar{z}^{(l-1)}$ in expression (2) is understood as (0,1)-rolled for the simplicity of notations. All products in this paper given below should be understood as (0,1)-rolled if it is not pointed separately.

The multilayer artificial feed-forward neural network is the consecutive connection of the separate layers. The number of layers of the multilayer artificial neural network we will denote L, and variable l in the previous expressions takes values $l=1,2,...,L$. Three-layer (L = 3) artificial neural network is shown in Fig. 2. The general expressions describing the artificial neural network with arbitrary number of layers L can be obtained on the example of three-layer neural network.

Since

$$\begin{cases}\bar{z}^{(1)}=\bar{f}^{(1)}(\bar{v}^{(1)})=\bar{f}^{(1)}(w^{(1)}\bar{x}+b^{(1)}),\\ \bar{z}^{(2)}=\bar{f}^{(2)}(\bar{v}^{(2)})=\bar{f}^{(2)}(w^{(2)}\bar{z}^{(1)}+b^{(2)}),\\ \bar{z}^{(3)}=\bar{f}^{(3)}(\bar{v}^{(3)})=\bar{f}^{(3)}(w^{(3)}\bar{z}^{(2)}+b^{(3)}),\end{cases} \tag{3}$$

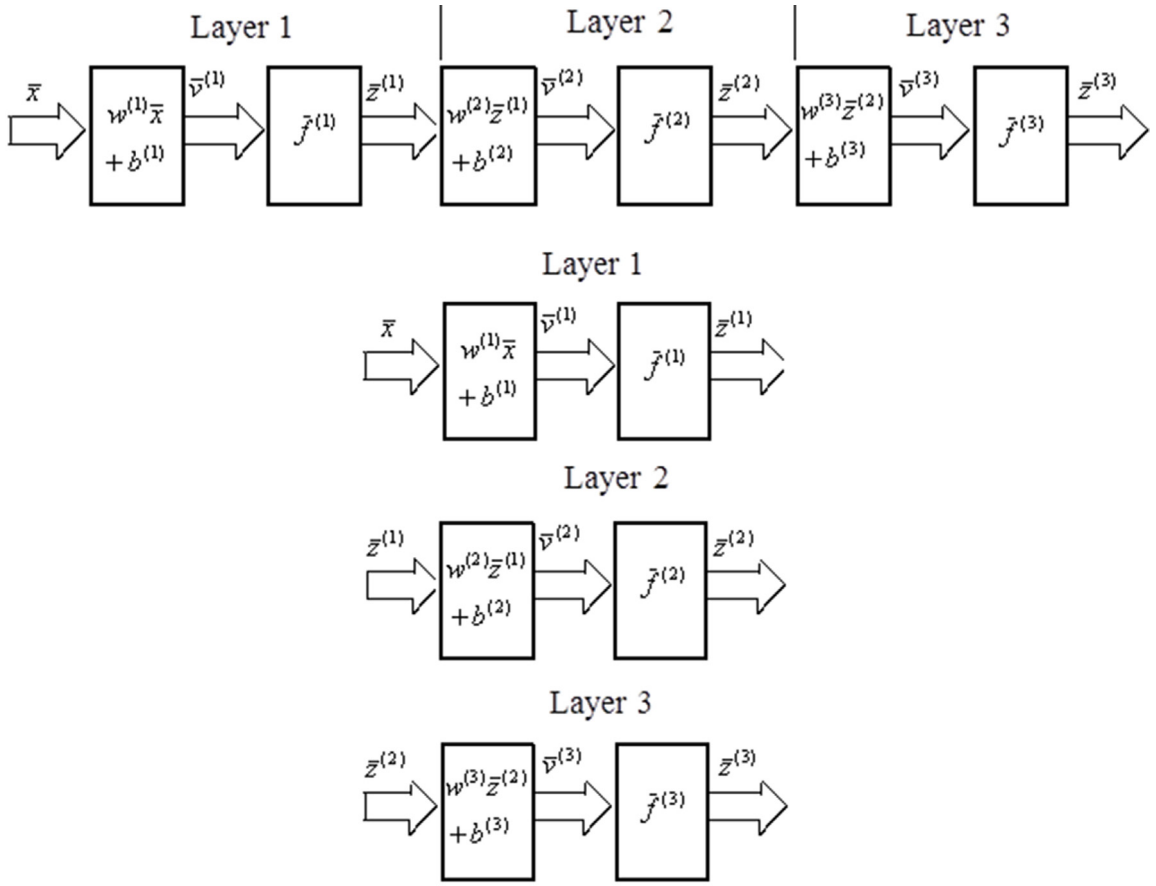

Fig. 2. Three-layer artificial neural network

then three-layer neural network is described by the following expression:

$$\overline{z}^{(3)} = \overline{f}^{(3)}\left(w^{(3)}\left(\overline{f}^{(2)}\left(w^{(2)}\left(\overline{f}^{(1)}(w^{(1)}\overline{x} + b^{(1)})\right) + b^{(2)}\right)\right) + b^{(3)}\right).$$

The following general formula is valid for L-layer artificial neural network instead of formula (3):

$$\overline{z}^{(l)} = \overline{f}^{(l)}(\overline{v}^{(l)}) = \overline{f}^{(l)}(w^{(l)}\overline{z}^{(l-1)} + b^{(l)}), \quad l = 1,2,...,L, \tag{4}$$

where $\overline{z}^{(0)} = \overline{x}$, $\overline{v}^{(l)} = w^{(l)}\overline{z}^{(l-1)} + b^{(l)}$, $l = 1,2,...,L$.

Learning of the artificial neural network consists of the selection of weight matrices $w^{(1)}, w^{(2)}, ..., w^{(L)}$ and bias vectors $\overline{b}^{(1)}, \overline{b}^{(2)}, ..., \overline{b}^{(L)}$ providing the best performance of its functions by the neural network. The learning set (the training set) contains learning pairs $(\overline{x}_k, \overline{y}_k)$, $k = 1,2,...,n$, where $\overline{x}_k$ is the input vector of the artificial neural network and $\overline{y}_k$ is the required output vector (the target vector) of the artificial neural network corresponding to the input vector $\overline{x}_k$, n is the size of the training set. The loss function C is introduced to the neural network learning, which depends in general case on the set $\overline{\overline{y}} = \{\overline{y}_1, \overline{y}_2, ..., \overline{y}_n\}$ of the target vectors $\overline{y}_k$ and the set $\overline{\overline{z}}^{(L)} = \left(\overline{z}_1^{(L)}, \overline{z}_2^{(L)}, ..., \overline{z}_n^{(L)}\right)$ of the output vectors $\overline{z}_k^{(L)}$ of the neural network: $C = C(\overline{\overline{y}}, \overline{\overline{z}}^{(L)})$. For instance, it is possible to use the quadratic loss function

$$C = \sum_{k=1}^{n} {}^{0,1}\left(\overline{z}_k^{(L)} - \overline{y}_k\right)^2. \tag{5}$$

We use the notation of (0,1)-rolled square of the matrix in expression (5) (in this case (0,1)-rolled square of the vector). For the task of the pattern recognition, $\overline{y}_k$ is the number of the pattern corresponding to the input vector $\overline{x}_k$, and $\overline{z}_k^{(L)}$ is the scalar ($s_L = 1$) on the output of the neural network.

The proposed neural network learning algorithm supposes the calculation of the derivatives of the loss function on vector and matrix arguments. Since such a differentiation and the differentiation of the superposition are not defined in classical vector-matrix mathematical approach we use a multidimensional-matrix differentiation theory [8]. This is the main reason to use the multidimensional-matrix mathematical approach in this paper.

We have the following derivatives for the neural network in Fig. 2:

$$\frac{\partial C}{\partial \overline{z}_k^{(2)}} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{z}_k^{(2)}} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} \frac{\partial \overline{v}_k^{(3)}}{\partial \overline{z}_k^{(2)}},$$

$$\frac{\partial C}{\partial \overline{z}_k^{(1)}} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{z}_k^{(2)}} \frac{d\overline{z}_k^{(2)}}{d\overline{z}_k^{(1)}} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} \frac{\partial \overline{v}_k^{(3)}}{\partial \overline{z}_k^{(2)}} \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} \frac{\partial \overline{v}_k^{(2)}}{\partial \overline{z}_k^{(1)}},$$

$$\frac{\partial C}{\partial \overline{z}_k^{(0)}} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{z}_k^{(2)}} \frac{d\overline{z}_k^{(2)}}{d\overline{z}_k^{(1)}} \frac{d\overline{z}_k^{(1)}}{d\overline{x}_k} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} \frac{\partial \overline{v}_k^{(3)}}{\partial \overline{z}_k^{(2)}} \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} \frac{\partial \overline{v}_k^{(2)}}{\partial \overline{z}_k^{(1)}} \frac{d\overline{z}_k^{(1)}}{d\overline{v}_k^{(1)}} \frac{\partial \overline{v}_k^{(1)}}{\partial \overline{x}_k}.$$

Since

$$\frac{\partial \overline{v}_k^{(l)}}{\partial \overline{z}_k^{(l-1)}} = \frac{\partial}{\partial \overline{z}_k^{(l-1)}} \left(w^{(l)} \overline{z}_k^{(l-1)} + \overline{b}^{(l)} \right) = w^{(l)},$$

Then, instead of previous expressions we have the derivatives

$$\frac{\partial C}{\partial \overline{z}_k^{(2)}} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} w^{(3)},$$

$$\frac{\partial C}{\partial \overline{z}_k^{(1)}} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} w^{(3)} \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} w^{(2)},$$

$$\frac{\partial C}{\partial \overline{z}_k^{(0)}} = \frac{\partial C}{\partial \overline{x}_0} = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} w^{(3)} \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} w^{(2)} \frac{d\overline{z}_k^{(1)}}{d\overline{v}_k^{(1)}} w^{(1)}.$$

Let us denote the derivative as follows:

$$\frac{\partial C}{\partial \overline{z}_k^{(l)}} = (\partial C \overline{z}^{(l)})_k, \quad l = 1, 2, ..., L.$$

We get for three-layer neural network

$$(\partial C \overline{z}^{(2)})_k = (\partial C \overline{z}^{(3)})_k \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} w^{(3)},$$

$$(\partial C \overline{z}^{(1)})_k = \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} w^{(3)} \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} w^{(2)} = (\partial C \overline{z}^{(2)})_k \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} w^{(2)},$$

$$(\partial C \overline{z}^{(0)})_k = (\partial C \overline{z}^{(1)})_k \frac{d\overline{z}_k^{(1)}}{d\overline{v}_k^{(1)}} w^{(1)}.$$

The following general recurrent formula is valid for L-layer neural network:

$$\begin{cases} (\partial C \overline{z}^{(L)})_k = \partial C / \partial \overline{z}_k^{(L)}, \\ (\partial C \overline{z}^{(l-1)})_k = (\partial C \overline{z}^{(l)})_k \dfrac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} w^{(l)}, \quad l = L, L-1, ..., 2, 1, \ k = 1, 2, ..., n. \end{cases} \tag{6}$$

We have the following derivative for the quadratic loss function (5):

$$(\partial C\overline{z}^{(L)})_k = \partial C / \partial \overline{z}_k^{(L)} = 2^{0,1}\left(\overline{z}_k^{(L)} - \overline{y}_k\right), \quad k = 1,2,...,n. \tag{7}$$

We have to know the derivatives of the loss function $C = C\left(\overline{z}_1^{(3)}, \overline{z}_2^{(3)}, ..., \overline{z}_n^{(3)}, \overline{y}_1, \overline{y}_2, ..., \overline{y}_n\right)$ on the weight matrices. They are defined by the following expressions:

$$\frac{\partial C}{\partial w^{(3)}} = \sum_{k=1}^{n} \frac{\partial C}{\partial \overline{z}_k^{(3)}} \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} \frac{\partial \overline{v}_k^{(3)}}{\partial w^{(3)}} = \sum_{k=1}^{n} (\partial C\overline{z}^{(3)})_k \frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}} \frac{\partial \overline{v}_k^{(3)}}{\partial w^{(3)}}, \tag{8}$$

$$\frac{\partial C}{\partial w^{(2)}} = \sum_{k=1}^{n} \frac{\partial C}{\partial \overline{z}_k^{(2)}} \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} \frac{\partial \overline{v}_k^{(2)}}{\partial w^{(2)}} = \sum_{k=1}^{n} (\partial C\overline{z}^{(2)})_k \frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}} \frac{\partial \overline{v}_k^{(2)}}{\partial w^{(2)}}, \tag{9}$$

$$\frac{\partial C}{\partial w^{(1)}} = \sum_{k=1}^{n} \frac{\partial C}{\partial \overline{z}_k^{(1)}} \frac{d\overline{z}_k^{(1)}}{d\overline{v}_k^{(1)}} \frac{\partial \overline{v}_k^{(1)}}{\partial w^{(1)}} = \sum_{k=1}^{n} (\partial C\overline{z}^{(1)})_k \frac{d\overline{z}_k^{(1)}}{d\overline{v}_k^{(1)}} \frac{\partial \overline{v}_k^{(1)}}{\partial w^{(1)}}, \tag{10}$$

where $(\partial C\overline{z}^{(3)})_k$, $(\partial C\overline{z}^{(2)})_k$, $(\partial C\overline{z}^{(1)})_k$ are calculated by formulae (6). It is possible to write for the derivatives (8), (9), (10), denoted as $\partial Cw^{(l)}$, the general formula

$$\partial Cw^{(l)} = \frac{\partial C}{\partial w^{(l)}} = \sum_{k=1}^{n} (\partial C\overline{z}^{(l)})_k \frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} \frac{\partial \overline{v}_k^{(l)}}{\partial w^{(l)}}, \quad l = 1,2,...,L, \quad k = 1,2,...,n,$$

or

$$\partial Cw^{(l)} = \frac{\partial C}{\partial w^{(l)}} = \sum_{k=1}^{n} (\partial C\overline{z}^{(l)})_k (\partial \overline{z}^{(l)} w^{(l)})_k, \quad l = 1,2,...,L, \quad k = 1,2,...,n, \tag{11}$$

where the notation is introduced

$$(\partial \overline{z}^{(l)} w^{(l)})_k = \frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} \frac{\partial \overline{v}_k^{(l)}}{\partial w^{(l)}}, \quad l = 1,2,...,L, \quad k = 1,2,...,n. \tag{12}$$

Let us note that $(\partial \overline{z}^{(l)} w^{(l)})_k$ is three-dimensional ($s_l \times s_l \times s_{l-1}$)-matrix.

It is necessary to know the derivatives $d\overline{z}_k^{(l)} / d\overline{v}_k^{(l)}$ and $\partial \overline{v}_k^{(l)} / \partial w^{(l)}$ to calculate the expressions $(\partial \overline{z}^{(l)} w^{(l)})_k$ (12).

At first, let us find the derivative $d\overline{z}_k^{(l)} / d\overline{v}_k^{(l)}$. Since $\overline{z}^{(l)} = \left(z_1^{(l)}, z_2^{(l)}, ..., z_{sl}^{(l)}\right)$ and $\overline{v}^{(l)} = \left(v_1^{(l)}, v_2^{(l)}, ..., v_{sl}^{(l)}\right)$ are vectors, then $\overline{z}^{(l)} = \overline{f}^{(l)}(\overline{v}^{(l)})$ is the vector function $\overline{f}^{(l)}(\overline{v}^{(l)}) = \left(f_1^{(l)}(v^{(l)}), f_2^{(l)}(v^{(l)}), ..., f_{sl}^{(l)}(v^{(l)})\right)$, moreover, this function is such, that each coordinate $v_i^{(l)}$ is transformed into coordinate $z_i^{(l)}$ by the corresponding coordinate activation function $z_i^{(l)} = f_i^{(l)}(v_i^{(l)})$. If the coordinate activation functions are the same, i. e. $f_1^{(l)}(v^{(l)}) = f_2^{(l)}(v^{(l)}) = ... = f_{sl}^{(l)}(v^{(l)}) = f^{(l)}(v^{(l)})$, then $z_i^{(l)} = f^{(l)}(v_i^{(l)})$. The derivative $d\overline{z}_k^{(l)} / d\overline{v}_k^{(l)}$ denoted as $(d\overline{z}^{(l)}\overline{v}^{(l)})_k$ is defined in such a case as the following diagonal ($s_l \times s_l$)-matrix:

$$\frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} = (d\overline{z}^{(l)}\overline{v}^{(l)})_k = (d\overline{z}^{(l)}\overline{v}_{i,j}^{(l)})_k = \mathrm{diag}\left(\frac{df(v_i^{(l)})}{dv_i^{(l)}}\right), \quad i, j = 1,2,...,s_l, \tag{13}$$

where s_l is the length of the output vector of the activation function $\overline{f}^{(l)}$ of the l-th layer. In particular, if the activation function has the form

$$f\left(v_i^{(l)}\right) = \frac{1}{1 + \exp\left(-v_i^{(l)}\right)}$$

(logistic activation function) then the derivative is defined by the expression

$$\frac{df\left(v_i^{(l)}\right)}{dv_i^{(l)}} = f\left(v_i^{(l)}\right)\left(1 - f\left(v_i^{(l)}\right)\right),$$

so that

$$\frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} = (d\overline{z}^{(l)}\overline{v}^{(l)})_k = \left(d\overline{z}^{(l)}\overline{v}_{i,j}^{(l)}\right)_k = \operatorname{diag}\left(f\left(v_i^{(l)}\right)\left(1 - f\left(v_i^{(l)}\right)\right)\right), \quad i = 1,2,...,s_l.$$

Let us find now the derivative $(\partial\overline{v}^{(l)}w^{(l)})_k = \partial\overline{v}_k^{(l)} / \partial w^{(l)}$. Since $\overline{v}_k^{(l)} = w^{(l)}\overline{z}_k^{(l-1)} + \overline{b}^{(l)}$, then we obtain by the rule of product differentiation [8]

$$\partial\overline{v}w^{(l)} = \frac{\partial}{\partial w^{(l)}}()\left(w^{(l)}\overline{z}^{(l-1)} + \overline{b}^{(l)}\right) = {}^{0,1}\left(\left(\frac{dw^{(l)}}{dw^{(l)}}\right)^{B_2}\overline{z}^{(l-1)}\right)^{H_2} + {}^{0,1}\left(w^{(l)}\frac{d\overline{z}^{(l-1)}}{dw^{(l)}}\right) =$$

$$= {}^{0,1}\left(\left(E(0,2)\right)^{B_2}\overline{z}^{(l-1)}\right)^{H_2} = {}^{0,1}\left(E(0,2)\overline{z}^{(l-1)}\right)^{H_2},$$

where $E(0,2)$ is (0,2)-identity matrix with the respective order and B_2, H_2 are the transpose substitutions of the type "forward" and "back" respectively [8]. Since

$$Q = {}^{0,1}\left(E(0,2)\overline{z}^{(l-1)}\right) = \sum_{j'=1}^{s} e_{i,j,k,j'}z_{j'}^{(l-1)} = \sum_{j'=1}^{s} e_{i,k}e_{j,j'}z_{j'}^{(l-1)} = (q_{i,j,k}) = \left(e_{i,k}z_j^{(l-1)}\right) =$$

$$= {}^{0,0}\left(E(0,1)\overline{z}^{(l-1)}\right)^{(E_1,B_{2,1})},$$

then

$$\partial\overline{v}w^{(l)} = \left({}^{0,0}\left(E(0,1)\overline{z}^{(l-1)}\right)^{(E_1,B_{2,1})}\right)^{H_2} = {}^{0,0}\left(E(0,1)\overline{z}^{(l-1)}\right)^{(B_{2,1},E_1)} =$$

$$= {}^{0,0}\left(E(0,1)^{B_{2,1}}(\overline{z}^{(l-1)})^{E_1}\right) = {}^{0,0}\left(E(0,1)\overline{z}^{(l-1)}\right),$$

where $E(0,1)$ is (0,1)-identity matrix with the respective order; E_1, $B_{2,1}$ are the transpose substitutions of the type "identical" and "forward" respectively [8]; $e_{i,j,k,j'}$ are the elements of matrix $E(0,2)$; $e_{i,j}$ are the elements of matrix $E(0,1)$ [8]. As a result, we obtain

$$(\partial\overline{v}^{(l)}w^{(l)})_k = {}^{0,0}\left(E(0,1)\overline{z}_k^{(l-1)}\right) = \left(e_{i,j}z_{m,k}^{(l-1)}\right) = \left(\partial\overline{v}^{(l)}w^{(l)}{}_{i,j,m}\right)_k = \begin{pmatrix} z_{m,k}^{(l-1)} & \text{when } i = j \\ 0 & \text{otherwise} \end{pmatrix}, \quad (14)$$

$$l = 1,2,...,L, \quad i,j = 1,2,...,s_l, \quad m = 1,2,...,s_{l-1}, \quad k = 1,2,...,n.$$

Let us note that $(\partial\overline{v}^{(l)}w^{(l)})_k = \partial\overline{v}_k^{(l)} / \partial w^{(l)}$ (14) is three-dimensional ($s_l \times s_l \times s_{l-1}$)-matrix. If $l = 1$ then $z_{m,k}^{(0)} = x_{m,k}$, $m = 1,2,...,s_0$, in formula (14) (for the first layer).

Analogously, we get the derivatives of the loss function $C = C\left(\overline{z}_1^{(3)}, \overline{z}_2^{(3)}, ..., \overline{z}_n^{(3)}\right)$ on the bias vectors $\overline{b}^{(l)}$ of the layers:

$$\frac{\partial C}{\partial \overline{b}^{(3)}} = \sum_{k=1}^{n}\frac{\partial C}{\partial \overline{z}_k^{(3)}}\frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}}\frac{\partial\overline{v}_k^{(3)}}{\partial\overline{b}^{(3)}} = \sum_{k=1}^{n}(\partial C\overline{z}^{(3)})_k\frac{d\overline{z}_k^{(3)}}{d\overline{v}_k^{(3)}}\frac{\partial\overline{v}_k^{(3)}}{\partial\overline{b}^{(3)}},$$

$$\frac{\partial C}{\partial \overline{b}^{(2)}} = \sum_{k=1}^{n}\frac{\partial C}{\partial \overline{z}_k^{(2)}}\frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}}\frac{\partial\overline{v}_k^{(2)}}{\partial\overline{b}^{(2)}} = \sum_{k=1}^{n}(\partial C\overline{z}^{(2)})_k\frac{d\overline{z}_k^{(2)}}{d\overline{v}_k^{(2)}}\frac{\partial\overline{v}_k^{(2)}}{\partial\overline{b}^{(2)}},$$

$$\frac{\partial C}{\partial \overline{b}^{(1)}} = \sum_{k=1}^{n} \frac{\partial C}{\partial \overline{z}_k^{(1)}} \frac{d\overline{z}_k^{(1)}}{d\overline{v}_k^{(1)}} \frac{\partial \overline{v}_k^{(1)}}{\partial \overline{b}^{(1)}} = \sum_{k=1}^{n} (\partial C \overline{z}^{(1)})_k \frac{d\overline{z}_k^{(1)}}{d\overline{v}_k^{(1)}} \frac{\partial \overline{v}_k^{(1)}}{\partial \overline{b}^{(1)}}.$$

Let us write the general formula for these derivatives denoted as $\partial C\overline{b}^{(l)}$:

$$\partial C\overline{b}^{(l)} = \frac{\partial C}{\partial \overline{b}^{(l)}} = \sum_{k=1}^{n} (\partial C \overline{z}^{(l)})_k \frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} \frac{\partial \overline{v}_k^{(l)}}{\partial \overline{b}^{(l)}}, \quad l = 1,2,...,L, \quad k = 1,2,...,n,$$

or

$$\partial C\overline{b}^{(l)} = \frac{\partial C}{\partial \overline{b}^{(l)}} = \sum_{k=1}^{n} (\partial C \overline{z}^{(l)})_k (\partial \overline{z}^{(l)} \overline{b}^{(l)})_k, \quad l = 1,2,...,L, \quad k = 1,2,...,n, \tag{15}$$

where the following notation is introduced:

$$(\partial \overline{z}^{(l)} \overline{b}^{(l)})_k = \frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} \frac{\partial \overline{v}_k^{(l)}}{\partial \overline{b}^{(l)}}.$$

Since $\partial \overline{v}_k^{(l)} / \partial b^{(l)} = E(0,1)$, where $E(0,1)$ is (0,1)-identical s_l-th order matrix, then formula (15) is simplified to the following form:

$$(\partial \overline{z}^{(l)} b^{(l)})_k = \frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}}. \tag{16}$$

2. Learning algorithm of the neural network. Let us formulate the learning algorithm of the multilayer artificial feed-forward neural network obtained on the base of the results above.

Setting the arbitrary initial values of the parameters $w_m^{(1)}, w_m^{(2)}, ..., w_m^{(L)}$, $\overline{b}_m^{(1)}, \overline{b}_m^{(2)}, ..., \overline{b}_m^{(L)}$, of the neural network, m is the number of the learning iteration.

1. Performing the operations of points 2–8 of the algorithm below for each input action $\overline{x}_k$, $k = 1,2,...,n$, of the training set with the memorization of their results.

2. Calculation of the outputs $\overline{z}_k^{(l)}$, $l = 1,2,...,L$, of the layers by the forward motion with the use of formulae (4):

$$\overline{z}_k^{(0)} = \overline{x}_k, \quad \overline{z}_k^{(l)} = \overline{f}^{(l)}\left(\overline{v}_k^{(l)}\right) = \overline{f}^{(l)}\left(w^{(l)} \overline{z}_k^{(l-1)}\right), \quad l = 1,2,...,L.$$

3. Calculation of the derivatives $(d\overline{z}^{(l)} \overline{v}^{(l)})_k = \frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}}$, $l = 1,2,...,L$, by the formulae (13):

$$(d\overline{z}^{(l)} \overline{v}^{(l)})_k = \frac{d\overline{z}_k^{(l)}}{d\overline{v}_k^{(l)}} = \left(dz_i^{(l)} v_j^{(l)}\right)_k = \operatorname{diag}\left(f\left(v_i^{(l)}\right)\left(1 - f\left(v_i^{(l)}\right)\right)\right), \quad i = 1,2,...,s_l, \quad l = 1,2,...,L.$$

They are two-dimensional ($s_l \times s_l$)-matrices.

4. Calculation of the derivatives $(\partial \overline{v}^{(l)} w^{(l)})_k = \frac{\partial \overline{v}_k^{(l)}}{\partial w^{(l)}}$, $l = 1,2,...,L$, by the formulae (14):

$$(\partial \overline{v}^{(l)} w^{(l)})_k = {}^{0,0}\left(E(0,1)\overline{z}_k^{(l-1)}\right) = \left(e_{i,j} z_m^{(l-1)}\right)_k = \left(\partial v_i^{(l)} w_{j,m}^{(l)}\right) = \left(\begin{cases} z_m^{(l-1)} & \text{at } i = j \\ 0 & \text{otherwise} \end{cases}\right),$$

$$l = 1,2,...,L, \quad i,j = \overline{1,s_l}, \quad m = \overline{1,s_{l-1}}.$$

They are three-dimensional ($s_l \times s_l \times s_{l-1}$)-matrices.

5. Calculation of the derivatives $(\partial \overline{z}^{(l)} w^{(l)})_k = \frac{\partial \overline{z}_k^{(l)}}{\partial w^{(l)}}$ by the formulae (12):

$$(\partial \bar{z}^{(l)} w^{(l)})_k = \frac{d\bar{z}_k^{(l)}}{d\bar{v}_k^{(l)}} \frac{\partial \bar{v}_k^{(l)}}{\partial w^{(l)}}, \quad l = 1,2,...,L.$$

They are three-dimensional ($s_l \times s_l \times s_{l-1}$)-matrices.

6. Calculation of the derivatives $(\partial C\bar{z}^{(l)})_k = \dfrac{\partial C}{\partial \bar{z}_k^{(l)}}$, $l = 1,2,...,L,$ by the form of the loss function (5) and by formulae (6), (7):

$$\begin{cases} (\partial C\bar{z}^{(L)})_k = \partial C / \partial \bar{z}_k^{(L)}, \\ (\partial C\bar{z}^{(l-1)})_k = (\partial C\bar{z}^{(l)})_k \dfrac{d\bar{z}_k^{(l)}}{d\bar{v}_k^{(l)}} w^{(l)}, \quad l = L, L-1,...,2, \quad k = 1,2,...,n, \end{cases}$$

$$(\partial C\bar{z}^{(L)})_k = \frac{\partial C}{\partial \bar{z}_k^{(L)}} = 2\left(\bar{z}_k^{(L)} - \bar{y}_k\right).$$

They are vectors with s_l components.

7. Calculation of the components $\partial C w^{(l)} = \partial C / \partial w^{(l)}$, $l = 1,2,...,L,$ of the gradient ∇C by formulae (11):

$$\partial C w^{(l)} = \frac{\partial C}{\partial w^{(l)}} = \sum_{k=1}^{n} (\partial C\bar{z}^{(l)})_k (\partial \bar{z}^{(l)} w^{(l)})_k, \quad l = 1,2,...,L.$$

They are two-dimensional ($s_l \times s_{l-1}$)-matrices.

8. Calculation of the components $\partial C\bar{b}^{(l)} = \partial C / \partial \bar{b}^{(l)}$, $l = 1,2,...,L,$ of the gradient ∇C by formulae (15), (16):

$$\partial C\bar{b}^{(l)} = \frac{\partial C}{\partial \bar{b}^{(l)}} = \sum_{k=1}^{n} (\partial C\bar{z}^{(l)})_k (\partial \bar{z}^{(l)} \bar{b}^{(l)})_k, \quad l = 1,2,...,L.$$

They are vectors with s_l components. We get the gradient $\nabla C(W,B)$, $W = \{w^{(1)}, w^{(2)},..., w^{(L)}\}$, $B = \{\bar{b}^{(1)}, \bar{b}^{(2)},..., \bar{b}^{(L)}\}$, as the result of points 7, 8. The loop for each input action $\bar{x}_k$, $k = 1,2,...,n,$ of the training set and a single learning iteration are finished here.

9. Organisation of the search for the minimum of the loss function $C(w^{(1)}, w^{(2)}, w^{(3)}, \bar{b}^{(1)}, \bar{b}^{(2)}, \bar{b}^{(3)})$ by the gradient descent method: we find a new set of the parameters W_{m+1}, B_{m+1} by the formulae

$$W_{m+1} = W_m - \alpha \cdot \nabla C(W_m, B_m), \quad B_{m+1} = B_m - \alpha \cdot \nabla C(W_m, B_m),$$

where α is the step of the search.

10. If $\| W_{m+1} - W_m \| < \varepsilon$, $\| B_{m+1} - B_m \| < \varepsilon$, $\varepsilon > 0,$ then we go to point 11, otherwhise we replace the matrices $w_m^{(1)}, w_m^{(2)},..., w_m^{(L)}$ with the obtained new matrices $w_{m+1}^{(1)}, w_{m+1}^{(2)},..., w_{m+1}^{(L)}$, replace the vectors $\bar{b}_m^{(1)}, \bar{b}_m^{(2)},..., \bar{b}_m^{(L)}$ with the obtained new vectors $\bar{b}_{m+1}^{(1)}, \bar{b}_{m+1}^{(2)},..., \bar{b}_{m+1}^{(L)}$ and go to point 1.

11. We accept the sets of the matrices $W_{m+1} = \left\{w_{m+1}^{(1)}, w_{m+1}^{(2)},..., w_{m+1}^{(L)}\right\}$ and vectors $B_{m+1} = \left\{\bar{b}_{m+1}^{(1)}, \bar{b}_{m+1}^{(2)},..., \bar{b}_{m+1}^{(L)}\right\}$ obtained in point 10 as the final parameters of the neural network and complete the learning process.

3. On the pattern recognition by the learned neural network. The goal of the neural network in the patterns recognition is to assign the integer number (number of pattern) to each pattern instance (to each set of the features). Such a mapping is discontinuous. Since the neural network performs the continuous mapping then the output variable of the neural network should be transformed for the recognition to the integer number.

4. Computer modelling. The developed algorithm is realized programmatically in the program complex Matlab as the single m-file-function. The quadratic loss function, linear activation function for the last layer and logistic activation function for all other layers were chosen. The algorithm for

three-layers neural network was checked in the tasks of the approximation of three-degree scalar polynomial of two variables and the discriminant function for the gaussian patterns recognition with two features. The following parameters of the neural network are used: $s_0 = s_1 = s_2 = 2,\ s_3 = 1.$ The algorithm has shown its efficiency on these tasks. However, it was not possible to obtain the satisfactory approximation of the polynomial by the neural network. At the same time, the problems typical for any search system have been identified. They are the difficulty of choosing the search step size of the initial values of the parameters, low convergence speed of the learning algorithm, and specific for the artificial neural networks problems with choosing the number of layers, sizes of layers, activation functions [9].

5. Instance of recognition of four gaussian patterns. The recognition of 4 Gaussian patterns each of them with 2 features is modelled. Two features are chosen only for the possibility of the graphical illustration. Each pattern (class) is described by the mathematical expectation vector and covariance matrix. The following mathematical expectations A_i and covariance matrices R_i, $i = 1,2,3,4,$ are used:

$$A_1 = (0 \quad 0), \quad R_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

$$A_2 = (2.5 \quad 3.5), \quad R_2 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix},$$

$$A_3 = (1 \quad 5), \quad R_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.5 \end{pmatrix},$$

$$A_4 = (3 \quad 1), \quad R_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.3 \end{pmatrix}.$$

As you can see, the patterns are distinct in both mathematical expectations and covariance matrices and their features are not correlated. The patterns were assumed to be equally probable, i.e. with the probabilities $p_i = 1/4$.

50 realizations were modelled for learning and testing samples.

Three-layers neural network was used for the recognition. The sizes of the layers are defined as follows: $s_0 = s_1 = s_2 = 2,\ s_3 = 1.$ Quadratic loss function (5), linear activation function for the last layer and logistic activation function for all other layers apart the first one were chosen. The initial values of the elements of the weight matrices $w^{(i)}$ and the bias vectors $\overline{b}^{(i)}$ were chosen as random numbers from normal distribution. Step α of the gradient method is equal to 0.0015. The minimal value of the loss function equal to 25.5 in 100 iterations and the increment of the loss function of the last two iterations equal to 0.67823 were reached. The percentage of the correct recognition of each pattern on the testing set is 94, 92, 98, 78 and on the training set is 98, 90, 92, 86. Such recognition can be evaluated as quite well.

Graphical illustration related to the recognition of Gaussian patterns by the neural network is presented in Fig. 3–9.

6. Instance of the computer modelling of approximation of the polynomial by the neural network. The approximation of scalar two-degree polynomial of two variables $y = f(x_1, x_2)$ by the neural network was modelled. Values of the polynomial on the grid 5 × 9 of variables (x_1, x_2) were used for learning the neural network, i. e. 45 values. Three-layers neural network was used for the approximation. The sizes of the layers are defined as follows: $s_0 = s_1 = s_2 = 2,\ s_3 = 1.$ Quadratic loss function (5), linear activation function for the last layer and logistic activation function for all other layers apart the first one were chosen.

The initial values of the elements of the weight matrices $w^{(i)}$ and bias vectors $\overline{b}^{(i)}$ were chosen as random numbers from normal distribution. Step α of the gradient method is equal to 0.0015. The minimal value of the loss function equal to 295 in 1000 iterations with irregular nature of convergence was reached.

The graphical illustration related to the approximation of scalar two-degree polynomial of two variables by the neural network is presented in Fig. 10–12. The approximation in Fig. 12 can be evaluated as bad.

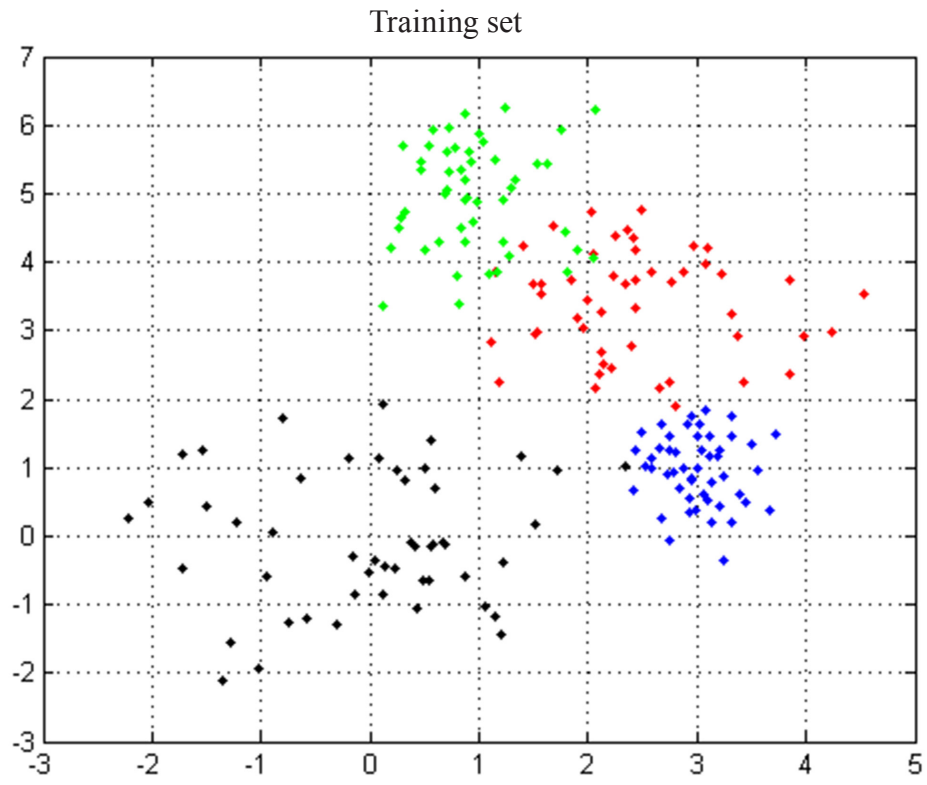

Fig. 3. The training set for the recognition of four Gaussian patterns

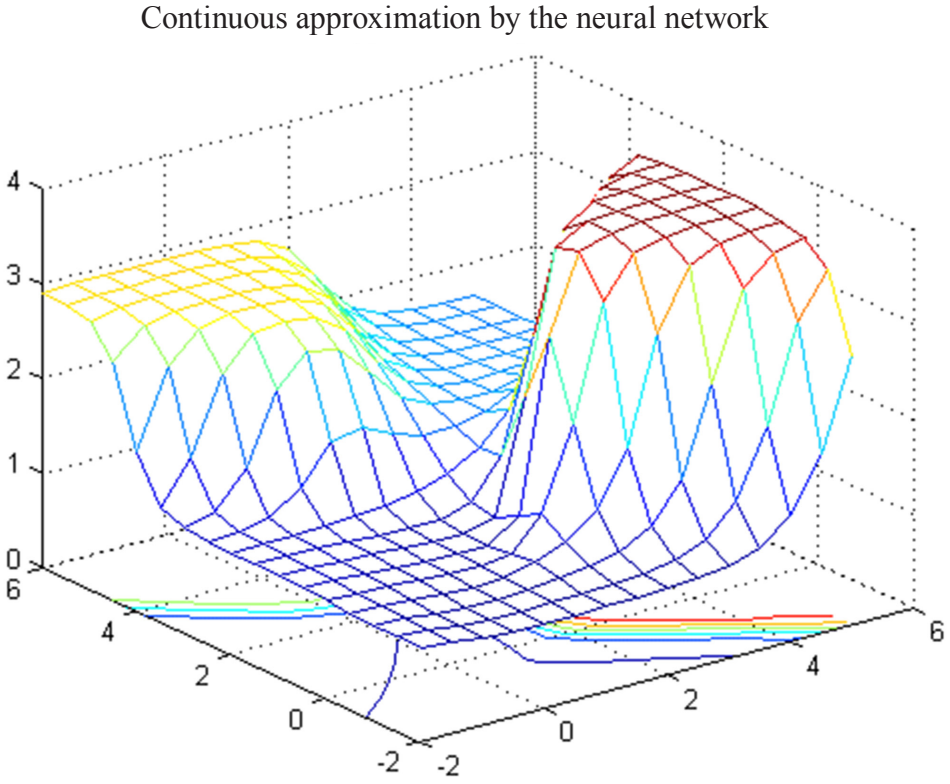

Fig. 4. Continuous approximation of the decision rule for recognition of four Gaussian patterns by the neural network

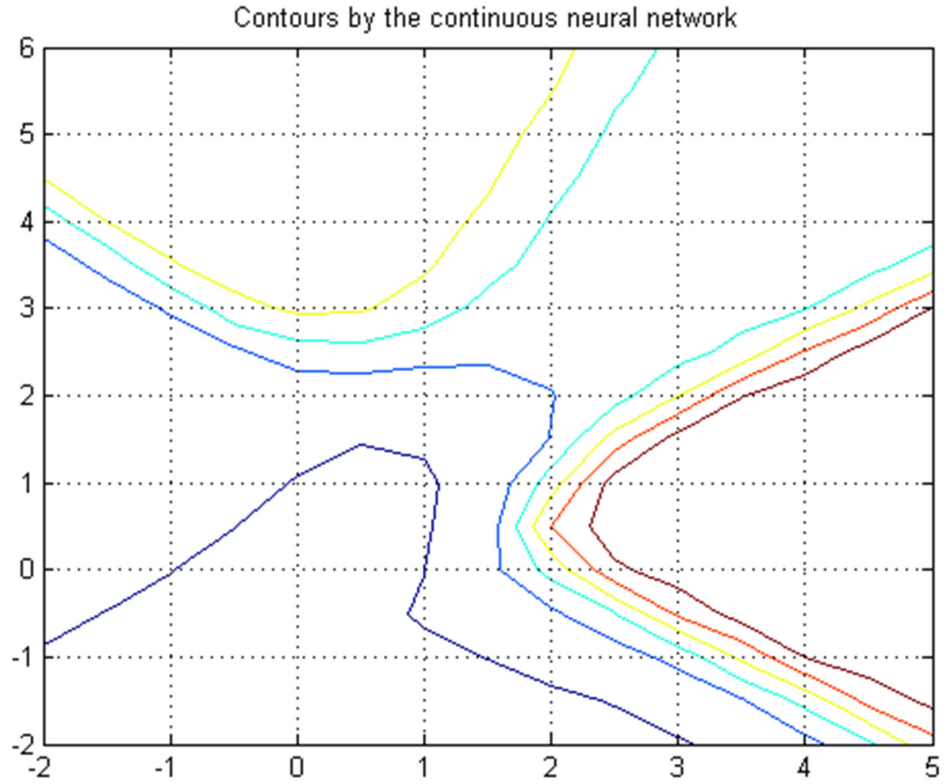

Fig. 5. Contours of the continuous approximation of the decision rule for recognition of four Gaussian patterns by the neural network

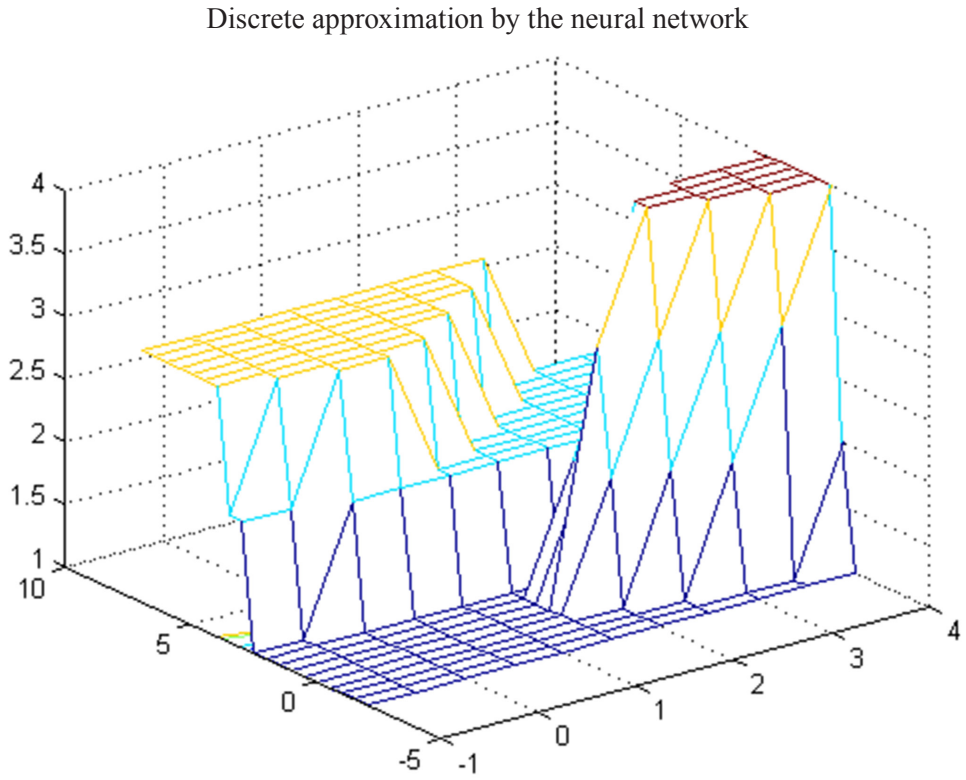

Fig. 6. Discrete approximation of the decision rule for recognition of four Gaussian patterns by the neural network

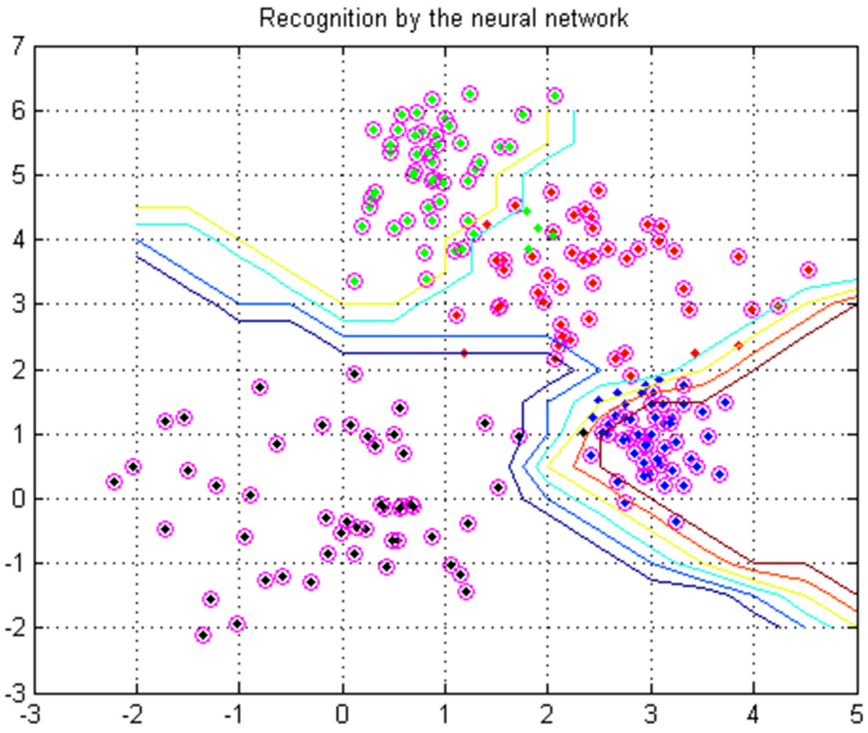

Fig. 7. Recognition of four Gaussian patterns by the neural network on the training set

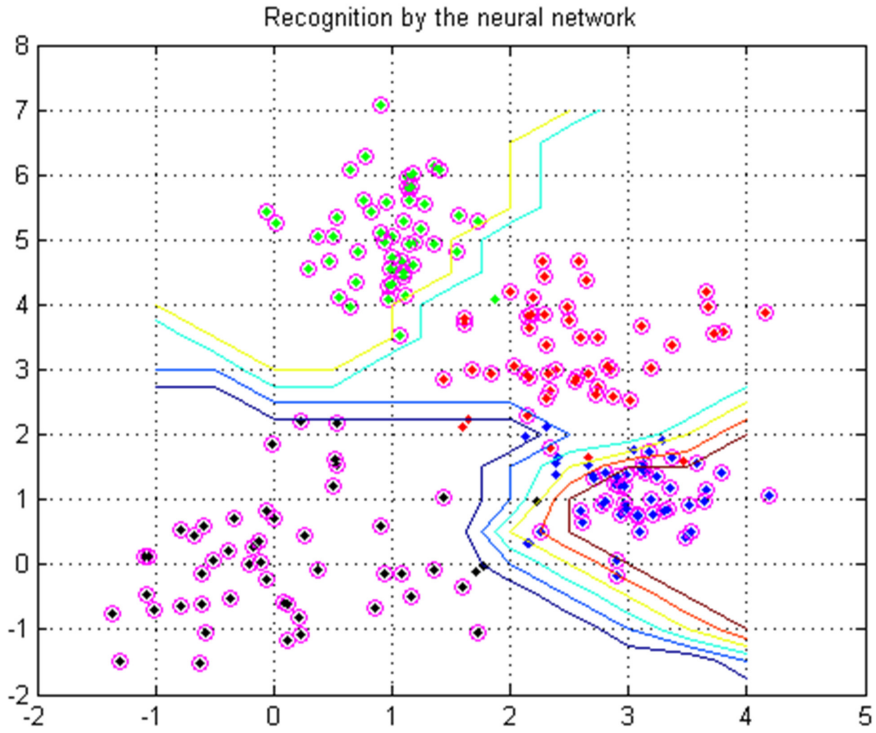

Fig. 8. Recognition of four Gaussian patterns by the neural network on the testing set

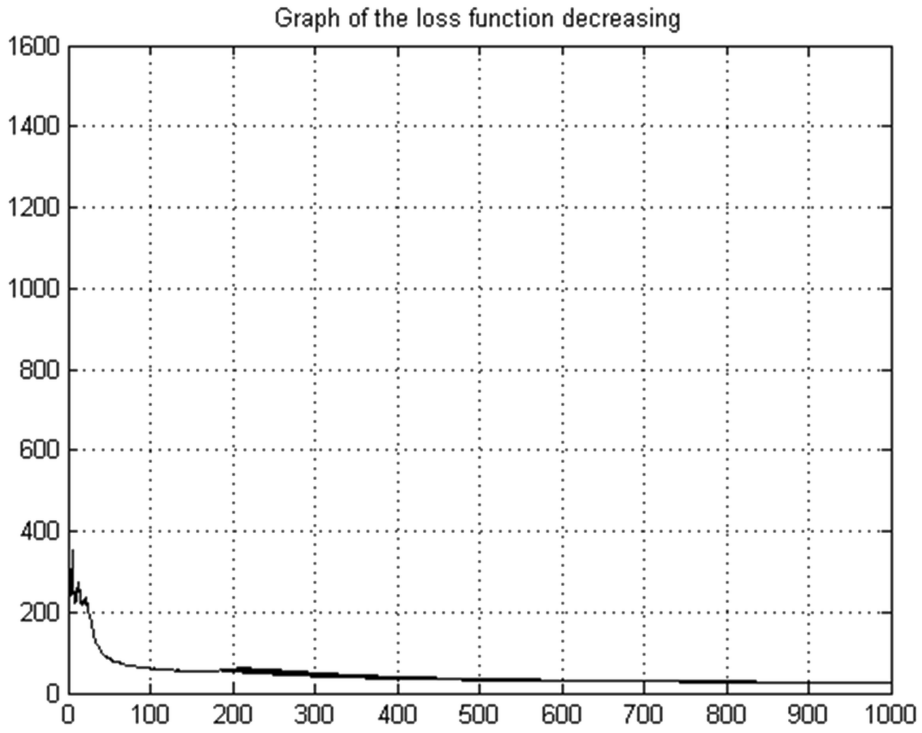

Fig. 9. Graph of the loss function decreasing in the learning process of the neural network

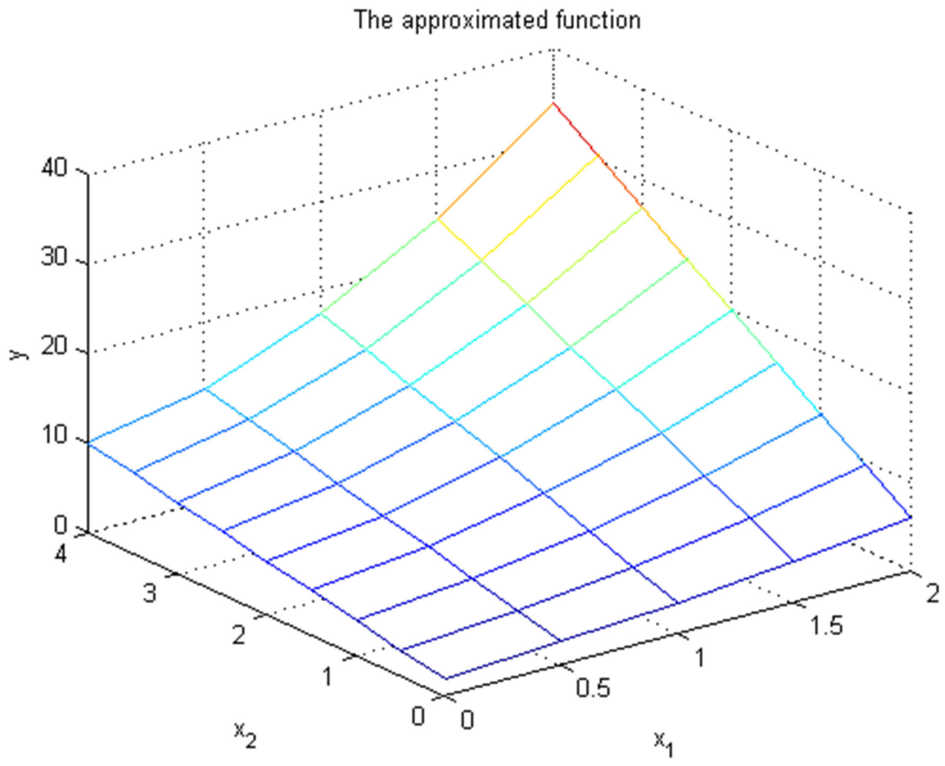

Fig. 10. Scalar two-degree polynomial of two variables subjected to the approximation by the neural network

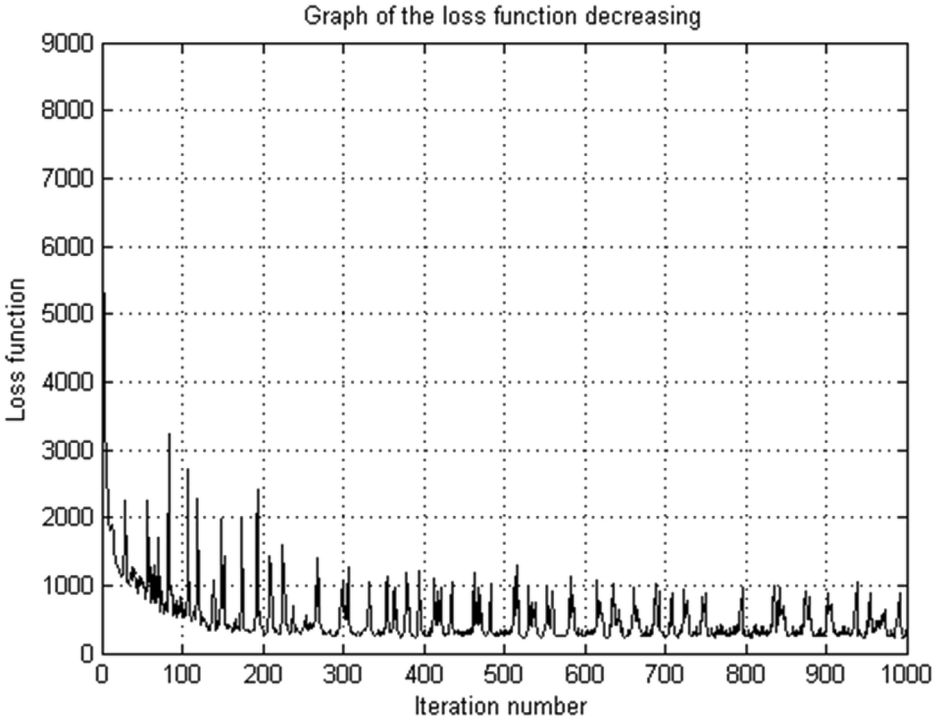

Fig. 11. Graph of the loss function decreasing in the learning process of the neural network

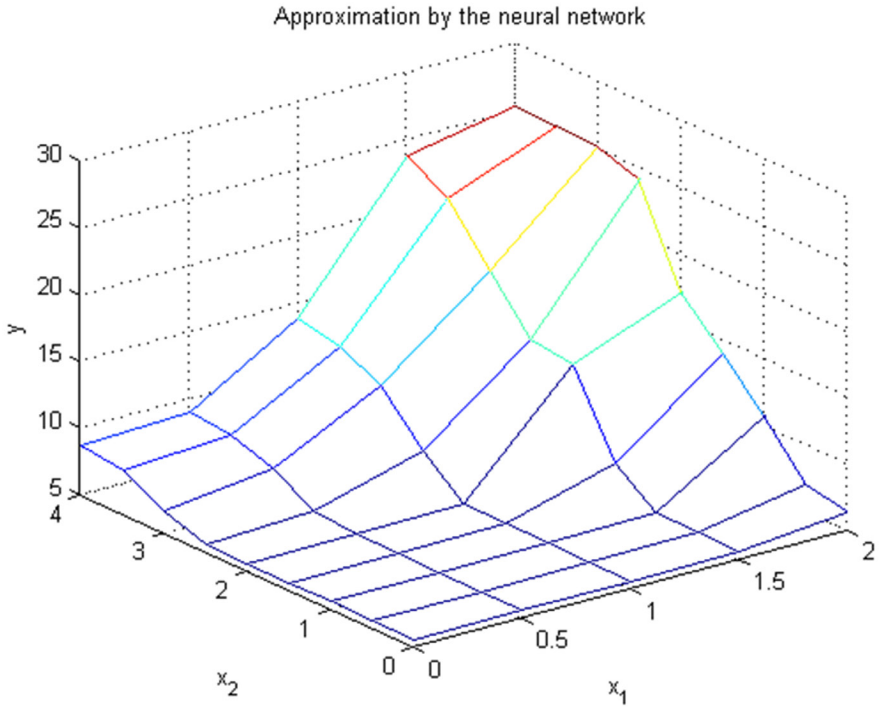

Fig. 12. Graph of the approximation of two-degree two-variables polynomial by the neural network

Conclusion. The main result of this work is the algorithm of the multilayer feed-forward neural network learning. A distinctive feature of the algorithm is its multidimensional-matrix form which ensures its applicability to neural networks with an arbitrary number of layers of an arbitrary size. In this sense,

it is suitable for the deep learning of neural networks [11]. The algorithm was realized programmatically in Matlab language as the single function. This program software confirmed its efficiency on the task of the approximation of polynomial of many variables and discriminant functions in patterns recognition. The program software is such that its multidimensional-matrix nature is not seen for the users since its parameters are defined in a usual matrix form. The disadvantages of traditional search algorithms as well as the specific advantages of the neural networks were exposed.

References

1. Golovinov A. O., Klimova E. N. Advantages of neural networks over traditional algorithms. *Eksperimental'nye i teoreticheskie issledovaniya v sovremennoi nauke: sbornik statei po materialam V Mezhdunarodnoi nauchno-prakticheskoi konferentsii* [Experimental and theoretical research in modern science: a collection of articles based on the materials of the V International scientific and practical conference]. Novosibirsk, Association of Researchers "Siberian Academic Book", 2017, pp. 11–15 (in Russian).

2. Mitrea C. A., Lee C. K. M., Wu Z. A Comparison between Neural Networks and Traditional Forecasting Methods: A Case Study. *International Journal of Engineering Business Management*, 2009, vol. 1, no. 2, pp. 19–24. https://doi.org/10.5772/6777

3. Blackard J. A., Dean D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 1999, vol. 24, no. 3, pp. 131–151. https://doi.org/10.1016/s0168-1699(99)00046-0

4. Eze C. M., Ugwuowo I. F., Asogwa O. A comparative analysis of vector autoregressive model and neural networks. *EPH – International Journal of Mathematics and Statistics*, 2018, vol. 4, no. 2, pp. 1–13. https://doi.org/10.53555/eijms.v4i2.21

5. Charef F., Ayachi F. A Comparison between Neural Networks and GARCH Models in Exchange Rate Forecasting. *International Journal of Academic Research in Accounting, Finance and Management Sciences*, 2016, vol. 6, no. 1, pp. 94–99. https://doi.org/10.6007/ijarafms/v6-i1/1996

6. Rumelhart D. E., Hinton G. E., Williams R. J. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol. 1.* Cambridge, MA, MIT Press, 1986, pp. 318–362. https://doi.org/10.7551/mitpress/4943.003.0128

7. Bishop C. M. *Pattern Recognition and Machine Learning.* Springer Science + Business Media, LLC, 2006. 738 p.

8. Mukha V. S. *Analysis of Multidimensional Data*. Minsk, Tekhnoprint Publ., 2004. 368 p. (in Russian).

9. Golovko V. A. *Neural Network: Learning, Organization and Application*. Moscow, IPRJP Publ., 2001. 256 p. (in Russian).

10. Schmidhuber J. Deep learning in neural networks: An overview. *Neural Networks*, 2015, vol. 61, pp. 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

11. Nielsen M. *Neural Networks and Deep Learning. Vol. 25*. San Francisco, CA, USA, Determination Press, 2015. Available at: https://jingyuexing.github.io/Ebook/Machine_Learning/Neural%20Networks%20and%20Deep%20Learning-eng.pdf (accessed 10 January 2025).

Information about the author

Vladimir S. Mukha – Dr. Sc. (Engineering), Professor, Professor of the Department of Information Technologies of Automated Systems, Belarusian State University of Informatics and Radioelectronics (6, P. Brovka Str., 220013, Minsk, Republic of Belarus). E-mail: mukhavladimir@gmail.com

Информация об авторе

Муха Владимир Степанович – доктор технических наук, профессор, профессор кафедры информационных технологий автоматизированных систем, Белорусский государственный университет информатики и радиоэлектроники (ул. П. Бровки, 6, 220013, Минск, Республика Беларусь). E-mail: mukhavladimir@gmail.com