DOI 10.1007/s10559-023-00522-0
Cybernetics and Systems Analysis, Vol. 58, No. 6, November, 2022

DECOMPOSITION METHOD FOR CALCULATING THE WEIGHTS OF A BINARY NEURAL NETWORK

A. Litvinenko,[1†] **D. Kucherov,**[1‡] **and M. Glybovets**[2]

UDC 519.816(045)

Abstract. *A method for determining the weights of a binary neural network based on its decomposition into elementary modules is presented. The approach allows tuning the weight coefficients of all the network connections at the stage of its designing, which eliminates the implementation of time-consuming iterative algorithms for training the network during its operation. An algorithm and an example of calculating the weights are given.*

Keywords: *binary neural network, weights, calculation method, decomposition, algorithm.*

INTRODUCTION

The widespread use of neural networks (NN) for solving various complex applied problems is due to modern advances in artificial intelligence, computing, and programming. The main advantage of using the NN models is their natural ability to self-learning and parallelization [1, 2].

Due to the large dimension of NN models, there is an accompanying number of their shortcomings. Among them, the following are traditionally distinguished: the use of heuristics in most design approaches, the duration of iterative adjustment of internal connections between basic elements, problems with finding real data to form a training sample, the possibility of obtaining a false solution, etc. These problems are partially solved by improving the structure of the NN and separating the processes of training and model operation in time [3–5].

This article proposes an approach to realization of a binary neural network whose inputs and outputs are the sets of bits, and neurons implement the functions of binary logic of several variables. The networks of this type are used to solve clustering problems in computer vision systems, telecommunication signal processing devices, disease diagnosis systems, etc. In real problems, the dimension of the input may be large.

The proposed approach is significantly different from constructing networks of known types. Better performance of NNs is expected due to the finite number of options for the complete enumeration of functions; therefore, the training is completed in a reasonable time and the hardware realization of the network is much simpler.

LITERATURE REVIEW

One of the main problems of designing an artificial NN is to calculate the weights of its connections [1–5]. To solve it, an approach is usually used that involves the realization of the procedure for “training” the NN [6, 7].

NN learning algorithms are based on two Hebb’s rules [3–6], which, for binary networks, can be formulated as follows:

— if the output signal of a neuron is incorrect and is equal to 0, the weight coefficients of its input connections through which the signal equal to 1 has passed must be increased;

[1]National Aviation University, Kyiv, Ukraine, [†]*litvinen@nau.edu.ua*; [‡]*d_kucherov@ukr.net*. [2]National University of “Kyiv-Mohyla Academy,” Kyiv, Ukraine, *glib@ukma.edu.ua*. Translated from Kibernetyka ta Systemnyi Analiz, No. 6, November–December, 2022, pp. 45–53. Original article submitted June 26, 2022.

— if the output signal of a neuron is incorrect and equal to 1, the weight coefficients of its input connections through which the signal equal to 1 has passed must be reduced.

The most common method of NN supervised training is the method of backpropagation of the error according to the delta rule [1–6].

For unsupervised NN training, the method of error correction according to the alpha rule is most often used [1–7]. This method involves a step-by-step increase in the weights of active NN connections by the same (pre-calculated) value while maintaining the same values for the weights of the connections that are not active.

Synthesis and hardware realization of neural networks are studied in [8, 9].

The possibility of constructing an NN on discrete elements with memory elements and binary input and output is described in [10] for modeling the life of a population of virtual bacteria developing in a bounded two-dimensional domain.

In [11], the training of a single-layer and multilayer neural networks with a short connection using binary classification is considered under the condition of achieving a zero training error in all the local minima with a correctly selected surrogate loss function.

The problem of binary classification is not trivial in case where the objects are satellite [10] or X-ray images [13, 14].

In [15], an attribute-based image retrieval scheme from CNN using an efficient small-size binary autoencoder and a nearest neighbor search is presented.

In [16], it is proposed to solve the problem of the dependence between the accuracy and training performance of a binary Bayesian classifier using a compromise. For this purpose, a risk function is introduced, in which the influence of one of the characteristics under study is compensated by the introduction of the corresponding Lagrange multiplier.

Note that all known NN training algorithms are iterative in nature and require significant machine time. However, for many applications, it is possible to calculate a priori weight coefficients of an NN at the design stage that makes it possible to avoid the time-consuming iterative training procedure during operation.

The purpose of this article is to present an approach to solving the problem of a priori calculation of the weights of a binary NN.

PROBLEM STATEMENT

Assume that a vector of signals $x = (x_i \mid i = \overline{1, n_1})$, $x \in X$, is an input at the classification NN, and the vector $y = (y_j \mid j = \overline{n - n_r + 1, n})$, $y \in Y$, of output signals is generated at the output, where X is the set of feasible input vectors; Y is the set of output vectors; n is the number of neurons in the NN; i, j are the numbers of neurons, $i = \overline{1, n}$, $j = \overline{1, n}$; r is the number of layers of the NN, which is determined by the requirements for the solution's accuracy.

It is also assumed that the NN is binary, i.e., the input and output signals of all neurons can take values from the set $\{0, 1\}$:

$$(\forall i = \overline{1, n_1})(x_i \in \{0, 1\});\ (\forall k = \overline{2, r})(i \in I_{k-1})(x_i \in \{0, 1\});\ (\forall j = \overline{1, n})(y_j \in \{0, 1\}),$$

where I_k is the set of neuron numbers belonging to the kth layer, $k = \overline{1, r}$; k is the number of a neural network's layer, $k = \overline{1, r}$; n_k is the number of neurons belonging to the kth layer, $k = \overline{1, r}$.

Activation functions $f_j(s_j)$ of all neurons are unit step functions with zero eliminations.

For neurons of the first (input) layer, $f_i(s_i) = x_i$, $i = \overline{1, n_1}$. For the neurons of all subsequent layers,

$$f_j(s_j) = \begin{cases} 0 & \text{for } s_j \le 0, \\ 1 & \text{for } s_j > 0, \end{cases}$$

where s_j is the weighted sum of input signals of the jth neuron, $s_j = \sum\limits_{i \in I_{k-1}} w_{ij} x_{ij}$, $j \in I_k$, $k = \overline{2, r}$. Here, x_{ij} is the input signal of the jth neuron coming from the ith neuron of the previous layer of the NN, $x_{ij} = y_i$, $i \in I_k$, $j \in I_{k+1}$, $k = \overline{1, r-1}$; w_{ij} is the the weight coefficient of the connection coming from the ith neuron to the jth neuron, $i \in I_k$, $j \in I_{k+1}$, $k = \overline{1, r-1}$.

If each possible input vector from the training sample of the NN is matched with the expected (reference) output vector, then the NN correctly classifies the input signal. Therefore, the quality criterion of a binary NN can be determined by the expression

$$\varepsilon = \frac{1}{L}\sum_{l=1}^{L}(y_l - y^*)^2 \le \varepsilon_{\text{feasible}},$$

where L is the number of the NN adjustment cycles; l is the number of adjustment cycle; ε is the root mean square error of NN classification based on the accepted training sample of the vectors $x \in X$; y_l is the vector of output signal values at the lth adjustment cycle; y^* is the vector of reference values for the output signals, $y^* \in Y$; $\varepsilon_{\text{feasible}}$ is a positive real number determined by the NN's designer that characterize a feasible deviation of the actual output signal from the reference one.

We need to calculate the vector of values $w = (w_{ij} \mid i \in I_k, j \in I_{k+1}, k = \overline{1, r-1})$ of weight coefficients such that, when a vector of binary signals $x \in X$ comes at the NN input, the network outputs the vector $y \in Y$ of correct values for the output binary signals that satisfy the criterion ε.

CALCULATION OF WEIGHT COEFFICIENTS

A peculiarity of the NN is a mutual independence of the weight coefficients of connections between the neurons. In other words, the weights of connections $w_{ij'}$ $(i \in I_k, j' \in I_{k+1})$ belonging to the j'th neuron are not connected by any mathematical relations with the weight coefficients $w_{ij''}$ $(i \in I_k, j'' \in I_{k+1}, j' \ne j'')$ of connections belonging to the j''th neuron $k = \overline{1, r-1}$. This makes it possible to decompose the NN into simple fragments (elementary modules) and solve the problem of calculating the weight coefficients for each module separately.

Let us explain the crux of the proposed method.

For each set of values $x = (x_i \mid i = \overline{1, n_1})$ of input signals, the set $y^* = (y_j^* \mid j = \overline{n - n_r + 1, n})$ of the appropriate reference (correct) values of output signals of the NN is established.

Next, the fragments (elementary modules) consisting of one neuron of the kth layer N_j, $j \in I_k$, and all the neurons of the previous $(k-1)$th layer N_i, $i \in I_{k-1}$, $k = \overline{1, r-1}$, are allocated on the network (Fig. 1).

Elementary modules are extracted sequentially starting with the initial layer of the NN ($k = r$) and ending with the second layer ($k = 2$).

Further, for each jth ($j = \overline{n - n_r + 1, n}$) output channel of the network, we separately calculate the values of the weight coefficients w_{ij}, $i \in I_{r-1}$, for connections going to the jth neuron from the neurons of the previous $(r-1)$th layer.

The weights w_{ij}, $i \in I_{r-1}$, must be such that the output signal of the jth neuron coincides with the desired reference value $y_j = y_j^*$.

The output signal y_j has a value of the activation function $f_j(s_j)$. Therefore, the activation function should take the value y_j^*, i.e., $f_j(s_j) = y_j^*$.

If $y_j^* = 0$, then the jth the neuron should not be activated; therefore, the weighted sum of its input signals should not exceed 0:

$$s_j = \sum_{i \in I_{r-1}} w_{ij} x_{ij} \le 0. \tag{1}$$

If $y_j^* = 1$, then the jth the neuron must be activated; therefore, the weighted sum of its input signals must be greater than 0:

$$s_j = \sum_{i \in I_{r-1}} w_{ij} x_{ij} > 0. \tag{2}$$

Expressions (1) and (2) allow us to formulate the conditions that the values of the weight coefficients w_{ij}, $i \in I_{r-1}$, must satisfy. To do this, we need to substitute all possible combinations of input signal values $x_{ij} \in \{0,1\}$, $i \in I_{r-1}$, into formulas (1) and (2) sequentially.

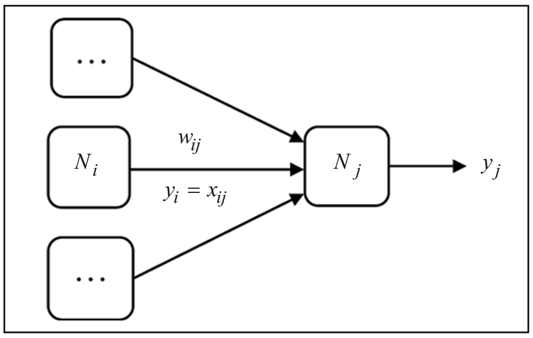

Fig. 1. Elementary module of an NN.

Let $x_j^* = (x_{ij}^* \mid i \in I_{r-1})$ be the vector of values of input signals coming to the input of the *j*th neuron, $j \in I_r$; $I_{r-1}^{(1)}$ be the set of neuron numbers of the $(r-1)$th layer, from which the signals equal 1 enter on the input of the *j*th neuron, $I_{r-1}^{(1)} = \{i \in I_{r-1} \colon x_{ij}^* = 1\}$; then conditions (1), (2), which must satisfy the values of the weight coefficients w_{ij}, $i \in I_{r-1}$, can be written in the form of the following inequalities:

$$\sum_{i \in I_{r-1}^{(1)}} w_{ij} \le 0 \ \text{ for } \ y_j^* = 0 \ \text{ and } \ \sum_{i \in I_{r-1}^{(1)}} w_{ij} > 0 \ \text{ for } \ y_j^* = 1.$$

For binary networks, it is enough to limit the range of values of the weight coefficients to the set $\{0, 1\}$. Thus, conditions (1), (2) can be written by the following implicit expressions:

$$(y_j^* = 0) \to \left(\sum_{i \in I_{r-1}^{(1)}} w_{ij} = 0 \right); \ (y_j^* = 1) \to \left(\sum_{i \in I_{r-1}^{(1)}} w_{ij} \ge 1 \right).$$

The inequality included in the second expression is satisfied by a set of combinations of the weight coefficients w_{ij}, $i \in I_{r-1}^{(1)}$. In order to avoid uncertainty, it is advisable to adopt a rule according to which, for $y_j^* = 1$, all weight coefficients are assigned the value 1. As a result, the rules for calculating weight coefficients w_{ij}, $i \in I_{r-1}^{(1)}$, of a binary network can be written as follows:

$$(y_j^* = 0) \to [(\forall i \in I_{r-1}^{(1)})(w_{ij} = 0)], \tag{3}$$

$$(y_j^* = 1) \to [(\forall i \in I_{r-1}^{(1)})(w_{ij} = 1)]. \tag{4}$$

The general formula for calculating the weight coefficients of a binary NN is

$$w_{ij} = \begin{cases} x_{ij} & \text{if } y_j^* = x_{ij}, \\ (1 - x_{ij}) & \text{if } y_j^* \ne x_{ij}, \end{cases} \tag{5}$$

$$y_j^* \in \{0, 1\}, \ x_{ij} \in \{0, 1\}, \ j = \overline{1, n}, \ i \in I_j,$$

where I_j is the set of neuron numbers from which the input signals of the *j*th neuron are received (in other words, the set of neuron numbers of the layer, which is the previous one for the layer, which contains the *j*th neuron).

Two conclusions follow from formula (5), which determine the necessary and sufficient conditions for obtaining the reference values for the output signal for the elementary module of the binary network:

— to obtain the value $y_j^* = 0$, it is required that the weights of all input connections of the *j*th neuron were equal to zero $(\forall i \in I_j)(w_{ij} = 0) \to (y_j^* = 0)$;

— to obtain the value $y_j^* = 1$, it is required that among the input signals of the jth neuron, there was at least one signal $x_{ij} = 1$ arriving on the input connection whose weight coefficient was greater than zero $(\exists i \in I_j)[(x_{ij} = 1) \& (w_{ij} > 0)] \to (y_j^* = 1)$.

After calculating the weights of the input connections of the j'th neuron from the rth layer, the described procedure is implemented for the next jth $(j \in J_r \setminus \{j'\})$ neuron of the output layer of the network.

After calculating the weights of the input connections of all neurons from the rth layer, elementary modules are formed, each of which consists of one neuron from the $(r-1)$th layer and all the neurons from the previous $(r-2)$th layer. Next, separately for each module, the values of the weight coefficients of connections between neurons from the $(r-2)$th and $(r-1)$th layers are calculated in the way presented above; then the previous couple of layers ($(r-3)$th and $(r-2)$th, etc.) are considered. Herewith, the output signal y_j $(j \in I_k)$ from each module is taken equal to 0 or 1 alternately and, as the input signals x_{ij}, $i \in I_{k-1}$, all possible binary combinations of their values are considered, $2 \le k \le r-1$.

The process ends with the calculation of the weight coefficients of the input connections between the neurons of the first and second layers of the NN. As a result, the full vector $w = (w_{ij} \mid i \in I_k, j \in I_{k+1}, k = \overline{1, r-1})$ of the weight coefficients, which ensures obtaining reference values for output signals for any combination of signals that can be received at the input of the NN, is determined.

During the network constructing, the additional input signal x_0 constructed to initialize the network when a zero vector $x = (x_i = 0 \mid i = \overline{1, n_1})$: $x_0 = \prod_{i=1}^{n_1} (1 - x_i)$ is received at its input should be provided.

ALGORITHM FOR CALCULATING WEIGHT COEFFICIENTS

The algorithm for calculating the weight coefficients of interneuron connections of NN involves the sequential execution of the following actions.

1. Fix the number j of the neuron N_j, which is considered as an output neuron for elementary module in the current iteration. The value j is chosen from the set $\{n - n_1 - 1, \ldots, n\}$ in descending order starting with n and ending with $(n - n_1 - 1)$.
2. Form the set I_j of neurons' numbers immediately preceding the neuron N_j.
3. Select the reference value for the output signal of the jth neuron y_j^* from the set $\{0, 1\}$.
4. Form the set $M_j^{(z)}$, $z \in \{0, 1\}$, of the vectors $x_i = (x_0, \ x_{ij} \mid i \in I_j)$ of values of input signals of the neuron N_j N_j at which it should produce the reference output signal y_j^*.
5. Calculate the weight coefficients w_{ij}, $i \in I_j$, of input connections of the neuron N_j by formula (5).
6. Check the conditions for ending of algorithm's cycles.

If the weights w_{ij}, $i \in I_j$, are not calculated for all combinations of input signals $x_i = (x_0, \ x_{ij} \mid i \in I_j)$, then go to Step 4.

If the weights w_{ij}, $i \in I_j$, are calculated for all combinations of input signals, but not for all reference values of the output signal $y_j^* \in \{0, 1\}$, then go to Step 3.

If the weights w_{ij}, $i \in I_j$, are calculated for all combinations of input signals and for all reference values of the output signal $y_j^* \in \{0, 1\}$, but not for all neurons N_j, $j \in \{n - n_1 - 1, \ldots, n\}$, then go to Step 1.

If the weights w_{ij}, $i \in I_j$, are calculated for all neurons N_j, $j \in \{n - n_1 - 1, \ldots, n\}$, then the algorithm ends.

The number of iterations required to calculate the weight coefficients of NN's connections is determined by the number of neurons $(n - n_1)$ that do not belong to the first (input) layer.

The proposed algorithm for calculating the weight coefficients of interneuron connections of an NN works accurately, and its time complexity is $O(\nu)$ and depends on the dimension of the vector of weight coefficients w, where ν is the number of NN's connections and $\nu = \sum_{k=1}^{r-1} n_k n_{k+1}$. The convergence of the given algorithm depends on the finiteness of the set of NN's elements and the absence of functional dependencies between the weight coefficients of its connections.

Since most of the operations provided by the algorithm are assignment operations, and comparison operations are used only to implement formula (5) once for each NN connection, it can be assumed that the computational complexity $O(\nu)$ of the algorithm is close to linear.

REALIZATION OF THE METHOD

The realization of the proposed method is illustrated by an example of calculating the weights of a binary network constructed to solve the clustering problem.

Assume that the characteristics of the clustering objects, taking into account the additional input signal x_0, are described by the binary vectors $x = (x_0, x_1, x_2, x_3)$, $x_i \in \{0, 1\}$, $i = \overline{1, 3}$.

We consider three classes of objects whose numbers correspond to the values of the elements of two-dimensional binary vector of the output signals $y = (y_4, y_5)$, $y_j \in \{0, 1\}$, $j = \overline{4, 5}$, in particular, $y = (1, 0)$ — first class, $y = (0, 1)$ — second class, and $y = (1, 1)$ — third class.

Assume that the objects whose characteristics are described by the input vectors $(1, 0, 0, 0)$, $(0, 0, 0, 1)$, and $(0, 0, 1, 0)$ belong to the first class; by the vectors $(0, 0, 1, 1)$, $(0, 1, 0, 0)$, and $(0, 1, 0, 1)$ belong to the second class; by the vectors $(0, 1, 1, 0)$ and $(0, 1, 1, 1)$ belong to the third class.

We need to calculate the weights of the connections of the NN constructed to solve the clustering problem in the above formulation. For this purpose, an NN is formed, consisting of two layers and having three input and two output channels (Fig. 2).

On this network, two elementary modules can be distinguished. The first of them consists of the output neuron N_4 and the input layer neurons N_1, N_2, and N_3; the second consists of the output neuron N_5 and the same input neurons N_1, N_2, and N_3.

The algorithm for calculating the weight coefficients of neural connections of this NN involves the following sequential steps.

1. Fix the number of the neuron, which is considered as an output neuron of the elementary module at this iteration, $j = 5$.

2. Form the set I_5 of neurons' numbers that immediately precede the neuron N_5, $I_5 = \{1, 2, 3\}$.

3. Select the reference value of the neuron N_5 output signal from the set $\{0, 1\}$, $y_5^* = 0$.

4. Form the sets $M_5^{(0)}$ of vectors $x_i = (x_0, x_{i5} \mid i \in I_5)$ of values of input signals of the neuron N_5, at which it should produce a reference output signal $y_5^* = 0$:

$$M_5^{(0)} = \{(1, 0, 0, 0);\ (0, 0, 0, 1);\ (0, 0, 1\,0)\}.$$

5. Calculate the weight coefficients w_{15}, $i \in I_5$, of input connections of the neuron N_5 by formula (5), $w_{15} = w_{25} = w_{35} = 0$.

Execution of the algorithm for the reference value of the output signal $y_5^* = 1$ according to Steps 4 and 5:

— form the set $M_5^{(1)}$ of vectors $x_i = (x_0, x_{i5} \mid i \in I_5)$ of values of the input signals of the neuron N_5, at which it should produce a reference output signal $y_5^* = 1$,

$$M_5^{(1)} = \{(0, 0, 1, 1);\ (0, 1, 0, 0);\ (0, 1, 0, 1);\ (0, 1, 1, 0);\ (0, 1, 1, 1)\};$$

— calculate the weight coefficients w_{15}, $i \in I_5$, of input connections of the neuron N_5 by formula (5).

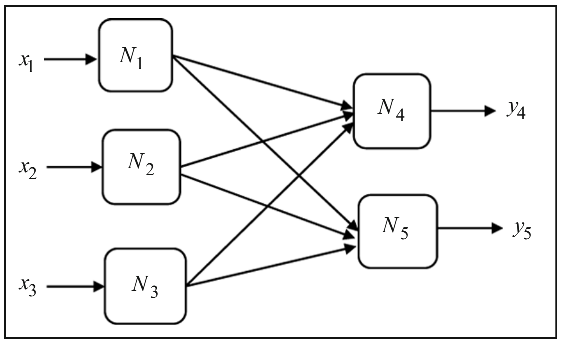

Fig. 2. The NN constructed to solve the clustering problem (Example).

Execution of the algorithm for the output neuron N_4 according to Steps 2–5:

— form the set I_4 of numbers of neurons that immediately precede the neuron N_4, $I_4 = \{1, 2, 3\}$;

— select the reference value for the output signal of the neuron N_4 from the set $\{0, 1\}$, $y_4^* = 0$;

— form the set $M_4^{(0)}$ of vectors $x_i = (x_0,\ x_{i4} \mid i \in I_4)$ of values of the input signals of the neuron N_4, at which it should produce a reference output signal $y_4^* = 0$:

$$M_4^{(0)} = \{(0, 0, 1, 1);\ (0, 1, 0, 0);\ (0, 1, 0, 1)\};$$

— calculate of weight coefficients w_{14}, $i \in I_4$, of input connections of the neuron N_4, by formula (5), $w_{14} = w_{24} = w_{34} = 0$.

Execution of the algorithm according to Steps 4 and 5 for the reference value of the output signal $y_4^* = 1$:

— form the set $M_4^{(1)}$ of vectors $x_i = (x_0,\ x_{i4} \mid i \in I_4)$ of values of the input signals of the neuron N_4, at which it should produce the reference output signal $y_4^* = 1$:

$$M_4^{(1)} = \{(1, 0, 0, 0);\ (0, 0, 0, 1);\ (0, 0, 1, 0);\ (0, 1, 1, 0);\ (0, 1, 1, 1)\};$$

— calculate the weight coefficients w_{14}, $i \in I_4$, of input connections of the neuron N_4 by formula (5), $w_{14} = w_{24} = w_{34} = 1$.

This completes the calculation process.

The class number of the classification object is determined by the binary code of the output signal.

The calculation of the weight coefficients requires four ($2n_r = 4$) iterations of the algorithm with the guarantee of absolute accuracy $\varepsilon = 0$ of the solving results.

To compare the proposed method with available approaches, the problem was solved using the Neuralnet function of the well-known machine learning language R. For this purpose, the set of possible input signal vectors was divided into three classes by the number of unit elements. The first class includes the vectors $(1, 0, 0, 0)$, $(0, 0, 0, 1)$, $(0, 0, 1, 0)$, and $(0, 1, 0, 0)$; the second class includes the vectors $(0, 0, 1, 1)$, $(0, 1, 0, 1)$, and $(0, 1, 1, 0)$; and the third class includes the vector $(0, 1, 1, 1)$. Each class is assigned an output signal $y_j \in \{0, 1\}$, $j = \overline{1, 3}$, whose unit value indicates that the input object belongs to a particular class.

The following results of weight coefficients calculation are obtained: $w_{01} = -6.05332$; $w_{02} = 3.74006$; $w_{03} = 0.28997$; $w_{11} = -10.83074$; $w_{12} = 5.82474$; $w_{13} = 5.46058$; $w_{21} = -6.90986$; $w_{22} = -7.97071$; $w_{23} = 5.2247$; $w_{31} = 7.55311$; $w_{32} = 3.56792$; $w_{33} = -0.45662$.

Mean square error of classification $\varepsilon = 0.543122$.

The number of network configuration cycles $L = 106$.

Comparison of the obtained results demonstrates the advantage of the proposed method of a priori calculation of the weight coefficients of binary NNs over traditional methods of their training for many applied areas.

CONCLUSIONS

The problem of calculating the weight coefficients of an NN is multivariate and has a finite (in the case of discrete signal values) or infinite (when the signals are measured by arbitrary real numbers) set of feasible solutions.

This property of the problem is explained by the mutual independence of the weight coefficients of the connections going to the inputs of the neurons of one layer from the neurons of the previous layer of the network. This makes it possible to decompose the general problem of calculating the weight coefficients of the NN into a number of subproblems solved separately for elementary fragments (modules) consisting of a single output neuron and a set of neurons of the previous layer, which is considered as the input layer of the module.

Using this method makes it possible to calculate the weight coefficients of the NN at the design stage and avoid the time-consuming iterative training procedure during operation. Moreover, the absolute correspondence of the output signals of the NN to their reference values for all predetermined input signals is achieved.

The method is implemented for binary NNs, but can be extended to networks of other classes. For example, when the activation functions have a more complex structure with an arbitrary offset, and the signals are measured by real numbers. In this case, expression (5) takes a different form.

Available methods of training artificial NNs are reduced to the implementation of heuristic algorithms that have all the disadvantages inherent in algorithms of this class. They are the following: lack of guarantees of finding a solution to the problem where it objectively exists; high probability of the search process getting into deadlocks that require human intervention (especially in problems that require compliance with conditions in the form of equations); negligible probability of finding the optimal solution to the problem, etc.

Obviously, there are many problems solved by NNs that can be replaced by rather simple Boolean, production, or logic models [17], which do not require time-consuming iterative procedures for synthesizing the structure and training the network, but lead to the desired result with absolute accuracy.

The decision to use a particular model depends on the subject area under consideration and requires expert assessment.

REFERENCES

1. G. F. Luger, Artificial Intelligence. Structures and Strategies for Complex Problem Solving, Addison Wesley (2004).
2. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, Upper Saddle River, NJ (2003).
3. S. Khaykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, Upper Saddle River, NJ (1999).
4. R. Callan, The Essence of Neural Networks, Prentice Hall Europe (1999).
5. O. Sigeru, Kh. Marzuki, and Y. Rybiyah, Neuro-Control and its Application, Springer (1996).
6. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, The MIT Press, Cambridge, Massachusetts–London, England (2016).
7. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer-Verlag (2009).
8. V. N. Opanasenko and S. L. Kryvyi, "Synthesis of neural-like networks on the basis of conversion of cyclic Hamming codes," Cybern. Syst. Analysis, Vol. 53, No. 4, 627–635 (2017). https://doi.org/10.1007/s10559-017-9965-z.
9. A. V. Palagin, V. N. Opanasenko, and S. L. Kryvyi, "FPGA based hardware implementation of cyclic Hamming code transformations," in: Proc. of the 15th All-Russian Sci. Techn. Conf. "Neuroinformatics-2013," Part 3, NIYAU MIFI, Moscow (2013), pp. 203–212.

10. A. V. Kazantsev, “Visual data processing and action control using binary neural network,” Eight Intern. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS’07) (Santorini, Greece, June 6–8, 2007), IEEE (2007), pp. 23. https://doi.org//10.1109/WIAMIS.2007.90.
11. S. Liang, R. Sun, Y. Li, and R. Srikant, “Understanding the loss surface of neural networks for binary classification,” in: Proc. of the 35th Intern. Conf. on Machine Learning, PMLR, Vol. 80 (2018), pp. 2835–2843. URL: https://proceedings.mlr.press/v80/liang18a/liang18a.pdf.
12. M. Krinitskiy, P. Verezemskaya, K. Grashchenkov, N. Tilinina, S. Gulev, and M. Lazzara, “Deep convolutional neural networks capabilities for binary classification of polar mesocyclones in satellite mosaics,” Atmosphere, Vol. 9, Iss. 11, 426 (2018). https://doi.org/10.3390/atmos9110426.
13. J. A. Dunnmon, D. Yi, C. P. Langlotz, C. Ré, D. L. Rubin, and M. P. Lungren, “Assessment of convolutional neural networks for automated classification of chest radiographs,” Radiology, Vol. 290, No. 2, 537–544 (2018). https://doi.org//10.1148/radiol.2018181422.
14. S. Korolev, A. Safiullin, M. Belyaev, and Y. Dodonova, “Residual and plain convolutional neural networks for 3D brain MRI classification,” arXiv:1701.06643v1 [cs.CV] 23 Jan 2017. URL: https://arxiv.org/pdf/1701.06643.pdf.
15. A. K. Menon and R. C. Williamson, “The cost of fairness in binary classification,” in: Proc. of the 1st Conf. on Fairness, Accountability and Transparency, PMLR, Vol. 81 (2018), pp. 107–118.
16. A. Ferreyra-Ramírez, E. Rodríguez-Martínez, C. Avilés-Cruz, and F. Lypez-Saca, “Image retrieval system based on a binary auto-encoder and a convolutional neural network,” IEEE Latin America Transactions, Vol. 18, No. 11, 1925–1932 (2020). https://doi.org//10.1109/TLA.2020.9398634.
17. A. Litvinenko, “Algorithms for solution inference based on unified logical control models,” Cybern. Syst. Analysis, Vol. 56, No. 2, 187–194 (2020). https://doi.org/10.1007/s10559-020-00234-9.