Deep Learning of Neural Networks Using Genetic Algorithms

Serhii Lienkov[1], Sergey Shvorov[2], Oleksandr Sieliukov[3], Igor Tolok[4], Nataliia Lytvynenko[5] and Taras Davydenko[6]

[1,4,5]*Military Institute of Taras Shevchenko National University of Kyiv, 81 Lomonosova Str., Kyiv, 03189, Ukraine*
[2,6]*National University of Life and Environmental Sciences, 15 Heroiv Oborony Str., Kyiv, 03041, Ukraine*
[3]*Kyiv National University of Construction and Architecture, 31 Povitroflotskyi Av., Kyiv, 03037, Ukraine*

Abstract

The technologies of artificial intelligence (AI) are aimed at creating a "thinking machine", that is, a computer system with human-like intelligence. One of the current directions of intellectualization is the use of neural networks with the implementation of their deep learning. The paper analyzes modern approaches to learning neural networks and investigates the possibility of using genetic algorithms to solve the problems of deep learning of neural networks. The purpose of the paper is to develop the scientific and methodological foundations of learning neural networks using genetic algorithms. To achieve the goal, the following tasks were solved: the justification of the approach to learning neural networks using genetic algorithms was carried out and the task of optimizing the learning of neural networks using a genetic algorithm was solved using the example of forecasting the time series of the environmental temperature by the method of shortest descent. A biotechnical complex exposed to external disturbances (external temperature) was chosen as the object on that relevant research was conducted.

Keywords 1

Genetic algorithm, genes, chromosomes, optimal solution, deep learning, search domain, evolution time.

1. Introduction

As you know, an artificial neural network (ANN) is a set of interconnected neurons. The transfer (activation) functions of all neurons in the network are fixed, and the weights are parameters of the network and can change. Some neuron inputs are labeled as external inputs of the network, and some outputs are labeled as external outputs of the network [1]. In general, applying any numbers to the network inputs determines some set of numbers at the network outputs.

Thus, the work of the neural network is to transform the input vector X into the output vector Y, and this transformation is given by the weights of the network.

Almost any problem can be reduced to a problem solved by a neural network. At the same time, the construction of the ANN is solved in two stages:

1) choosing the type (architecture) of the network;
2) selection of weights (training) of the network.

At the first stage, the following should be determined:

- what neurons we want to use (number of inputs, transfer functions);
- how they should be connected to each other;
- what to take as network inputs and outputs.

MoMLeT+DS 2022: 4th International Workshop on Modern Machine Learning Technologies and Data Science, November, 25-26, 2022, Leiden-Lviv, The Netherlands-Ukraine
EMAIL: lenkov_s@ukr.net (S. Lienkov); sosdoc@i.ua (S. Shvorov); selukov@3d.ua (O. Sieliukov); igortolok72@gmail.com (I. Tolok); n123n@ukr.net (N. Lytvynenko); dtr55@ukr.net (T. Davydenko)
ORCID: 0000-0001-7689-239X (S. Lienkov); 0000-0003-3358-1297 (S. Shvorov); 0000-0001-7979-3434 (O. Sieliukov); 0000-0001-6309-9608 (I. Tolok); 0000-0002-2203-2746 (N. Lytvynenko); 0000-0003-0277-6892 (T. Davydenko)

At first glance, this problem seems unsolvable, but it isn't necessary to invent a neural network "from scratch" - there are dozens of different neural network architectures, and the effectiveness of many of them has been proven mathematically [2].

At the second stage, the most difficult task is solved – it's necessary to "teach" the selected network, i.e. choose such values of its weights, so that the network works as required. In those neural networks used in practice, the number of weights can be several tens of thousands, so learning is a complex process. For many architectures, special learning algorithms have been developed that allow to adjust the weights of the network in a certain way. However, for complex ANN architectures, solving this problem requires further research. Solving the problem of deep learning of neural networks is especially relevant.

2. Problem Formulation

As shown in [1,3], depending on the functions performed by neurons in the network, three types can be distinguished:

1) input neurons are neurons that receive an input vector encoding an input action or an image of the external environment; they usually don't carry out computational procedures, information is transferred from the input to the output of the neuron by changing its activation;

2) output neurons are neurons whose output values represent the output of the network;

3) intermediate neurons are neurons that form the basis of artificial neural networks.

In most neural models, the type of neuron is related to its placement in the network. If a neuron has only output connections, then it's an input neuron, and on the contrary, it's an output neuron.

However, there may be a case where the output of a topologically internal neuron is considered as part of the network output. In the process of functioning (evolution of the state) of the network, the input vector is transformed into the output vector, i.e., some processing of information is carried out.

The process of the neural network (NN) functioning and the set of actions it can perform mainly depends on the values of synaptic connections. That's why, having determined the structure of the network corresponding to the selected problem area, the optimization of the weighting coefficients in real time is an urgent task [2,4]. Its ability to function qualitatively and adequately depends on how effectively the NN training will be performed.

The result of adding r is:

$$r = \sum_{i=1}^{n} V_i \cdot x_i + f \quad (1)$$

where: r - result of addition;

V_i – synapse weight $(i = 1, \dots, n)$; f – displacement value;
x_i – component of the input vector (input signal) $(i = 1, \dots, n)$;
n - number of neuron inputs.

The experiments on learning neural networks have shown that known methods of local and global optimization (gradient, stochastic, Newton, Hessian, etc.) require a significant number of learning steps, are sensitive to the accuracy of calculations, require a significant number of additional variables, therefore, the search and development of new methods is an urgent task learning of neural networks [5-7].

Therefore, there is a need to use such approaches, that wouldn't have the mentioned disadvantages. The genetic algorithm (GA) stands out among optimization mathematical devices in the context of the given task.

The idea of GA was expressed by J. Holland in the late 1960s and early 1970s.

The basic (classical, elementary or simple) genetic algorithm consists of the following steps:

1) selection of the initial population of chromosomes;
2) assessment of the fitness of chromosomes in the population;
3) checking the condition of stopping the algorithm;
4) selection of chromosomes;
5) application of genetic operators;
6) formation of a new population;

7) selection of the "best" chromosome.

The formation of the initial population consists in the random selection of a given number of chromosomes (individuals), that are represented by binary sequences of a fixed length (that is, the alleles of all genes in the chromosome are equal to 0 or 1).

The assessment of chromosome fitness in a population consists in calculating the fitness function for each chromosome of this population. The greater the value of this function, the higher the "quality" of the chromosome. The form of the fitness function depends on the nature of the problem tobe solved. It's assumed that the fitness function always takes non-negative values and, in addition, that this function must be maximized to solve the optimization problem. If the original form of the fitness function doesn't satisfy these conditions, then an appropriate transformation is performed (for example, the function minimization problem can be easily reduced to a maximization problem).

The determination of the stopping condition of the genetic algorithm depends on its specific application. In optimization tasks, if the maximum (or minimum) value of the fitness function is known, then the algorithm can stop after reaching the expected optimal value, possibly with a given accuracy [6-8]. Stopping of the algorithm can also happen if its execution doesn't lead to an improvement of the already achieved value. The algorithm can be stopped after a certain execution time or after performing a given number of iterations. If the stopping condition is met, the transitionto the final stage of selecting the "best" chromosome is made. Otherwise, the selection is performed inthe next step.

The chromosome selection consists in choosing (based on the values of the fitness function calculated at the second stage) those chromosomes that will participate in the creation of offspring for the next population, that is, for the next generation. Such a selection is made according to the principle of natural selection, according to that chromosomes with the highest values of the fitness function have the greatest chance of participating in the creation of new individuals. There are different methods of selection. The most popular is the so-called roulette method, that got its name by analogy with the well-known gambling game.

As a result of the selection process, a parent population (parental pool) is created with a number N equal to the number of the current population. The application of genetic operators to chromosomes selected by selection leads to the formation of a new population of descendants from the parent population created in the previous step.

Application of genetic operators. In the classic genetic algorithm, two basic genetic operators are used: the crossover operator and the mutation operator. However, it should be noted that the mutation operator plays a secondary role compared to the crossover operator. Since crossover in the classic genetic algorithm occurs almost always, while mutation is quite rare.

Formation of a new population. Chromosomes obtained as a result of the application of genetic operators to the chromosomes of the temporary parent population are included in the composition of the new population. It becomes the current population for this iteration of the genetic algorithm. At each subsequent iteration, the values of the fitness function are calculated for all chromosomes of this population, after that the condition for stopping the algorithm is checked and either the result is fixed in the form of the "best" chromosome (that has the largest value of the fitness function), or the transition is made to the next step of the genetic algorithm, i.e. to selection. In the classical genetic algorithm, the entire previous population of chromosomes is replaced by a new population of descendants having the same number.

In the classical genetic algorithm, only the binary coding method is used: selection by the "roulette wheel" method and point crossing (with one crossing point). To increase the efficiency of its work, many modifications of the basic classical genetic algorithm have been created, that are associated with the use of other selection methods, with the modification of genetic operators (first of all, the crossover operator), with the transformation of the fitness function, as well as with other ways of encoding the parameters of the problem in the form chromosomes.

These algorithms simulated evolutionary processes in the generations of such chromosomes. They implemented mechanisms of selection and reproduction similar to those used in natural evolution. Just as in nature, genetic algorithms searched for "good" chromosomes without using any information about the nature of the problem being solved. It was needed some estimate of each chromosome reflecting its fitness. The mechanism of selection consists in choosing chromosomes with the highest score (that is, the most adapted), that reproduce more often than individuals with a lower score (worse

adapted). The reproduction means the creation of new chromosomes as a result of the recombination of the genes of the parental chromosomes. The recombination is a process that results in new combinations of genes. Two operations are used for this: crossing over, that allows creating two completely new offspring chromosomes by combining the genetic material of a pair of parents, and mutation, that can cause changes in individual chromosomes.

Genetic algorithms use a number of terms borrowed from genetics, primarily genes and chromosomes, as well as population, individual, allele, genotype, phenotype.

Genetic algorithms are used in software development, in artificial intelligence systems, optimization, artificial neural networks and in other fields of knowledge. It should be noted that with their help problems are solved for that only neural networks were previously used. In this case, the genetic algorithms act simply as an alternative method independent of neural networks, designed to solve the same problem. The genetic algorithms are often used in conjunction with neural networks. They can support neural or jointly interact within the framework of a hybrid system designed to solve a specific task. The genetic algorithms are also used in conjunction with fuzzy systems.

The genetic algorithm is a method that reflects the natural evolution of problem solving methods, and primarily optimization problems. The genetic algorithms are search procedures based on mechanisms of natural selection and heredity. They use the evolutionary principle of survival of the fittest. Genetic algorithms differ from traditional optimization methods in the following main properties:

In terms of the speed of determining the optimal value of the objective function, the genetic algorithms are several orders of magnitude ahead of random search. However, the genetic algorithms aren't the only way to solve optimization problems. In addition to it, there are two main approaches to solving such problems - exhaustive and local-gradient, each of that has its advantages and disadvantages.

The iterative method is the easiest to program. To search for the optimal solution, it's necessary to calculate consistently the value of the objective function at all possible points, remembering the maximum (or minimum) of them. The disadvantage of the method is the high computational complexity, however, if it's possible to go through all options in a reasonable time, then the solution found is optimal.

The second approach is based on the gradient descent method. First, some random parameter values are chosen, and then these values are gradually changed, achieving the highest growth rate of the objective function. When a local maximum (minimum) is reached, this method stops, so additional measures are required to find the global optimum.

The gradient methods work quickly, but don't guarantee the optimality of the solution found. They are ideal for solving unimodal problems, where the objective function has a single local optimum (global). The practical problems, as a rule, are multimodal and multidimensional, for them there are no universal methods that allow to find quickly absolutely accurate solutions. By combining the screening and gradient methods, it can to get approximate solutions, the accuracy of that will increase as the calculation time increases.

The paper [9] shows the main differences between GA and standard optimization algorithms:

- the search for a suboptimal solution, based on the optimization of randomly given set of solutions, rather than one solution, that allows simultaneous analysis of several ways of approaching the extremum; evaluation of such solutions at each step allows synthesizing new solutions on the basis of old ones, i.e. evolutionary development of optimal solutions takes place;
- the solutions are considered as some coded structures, and not as a set of parameters, that allows in some cases to significantly reduce the time of data transformation, i.e. to increase the speed of finding optimal solutions;
- to assess the "suitability" of a decision for further evolutionary development, along with the use of the objective function, the "Rules of Survival" are additionally modeled, that increase the diversity of the set of decisions and determine the evolutionary development;
- the initialization, transformation and other types of decision operations use probabilistic rather than deterministic rules that introduce elements of randomness into the genetic search; therebysolving the problem of leaving local optima;
- there is no need to calculate the derivatives of the target function (as in gradient methods) or the matrix of derivatives of the second order (as in quasi-Newtonian methods);
- non-critical to the number of components of the admissible solution vector.

The authors of [4-6] consider the issue of operator adaptation in evolutionary computations and its application to optimize the structure of neural networks, effective multi-objective search for neural architecture using Lamarck evolution, such works indicate the need for further research, and especially, in

the direction of the application of genetic algorithms for learning neural networks.

The purpose of the paper is to develop the scientific and methodological foundations of learning neural networks using genetic algorithms.

To achieve the goal, the following tasks were solved:

1. The justification of the approach to learning neural networks using genetic algorithms was carried out;
2. The optimization problems of learning using the genetic algorithm on the example of forecasting the time series of the temperature of the natural environment by the method of shortest descent are solved.

3. Research materials and methodology

In the classical formulation, the task of learning neural network is considered as the task of finding the minimum of the learning error, that depends on the parameters of the network [7,8]. The quality of learning directly affects the prediction capabilities of the neural network, and therefore the accuracyof the problems being solved. Taking into account the above, it's possible to determine the network structure that corresponds to the chosen problem. The optimization of weighting factors in real time is relevant. The adequacy of its functioning depends on this.

The main functional purpose of an artificial neural network is the transformation of input signals (some scattered information about the external environment) into output signals (concepts about the external environment). Based on (1), the neural network in this case is represented as some multidimensional function $F: X \to Y$.

If the set of weight coefficients W_i of the input signals of the neurons of the network isn't ordered, then the function F implemented by the network is arbitrary. The set of all weights of all neurons corresponds to the vector W. The set of vectors W forms the state space of the neural network. Let us correspond to the initial state of the network with some arbitrary vector W^0. Then the trained neural network corresponds to W^*, i.e. such a state in that the one-valued mapping is realized $F: X \to Y$. In this case, the task of learning NN is formally reduced to the task of transition from some initial stateof the network corresponding to W^0, to the final state corresponding to W^*.

When learning a neural network, the problem of error minimization is set

$$\alpha(W) = \sum_{i=1}^{p} (z_i - b_i) \qquad (2)$$

where z_i - the value of the jth output of the neural network, is the known value of the jth output;
p - the number of neurons in the output layer.

If the network doesn't make errors, then $\alpha = 0$, that is, the goal of learning neural network is the task of finding the minimum of the error function (2) in the state space W.

In order to increase the reliability of decisions made on the basis of a neural network, it's necessary to investigate alternative optimization algorithms that allow finding the global extremum. Therefore, there is a need to use such approaches, that wouldn't have the indicated disadvantages. The considered genetic optimization algorithms are the most promising in this regard [7-8].

To perform the optimization procedure using the genetic algorithm, it's necessary:

1) to choose a representation of optimization parameters in the form of a certain data format: line, vector, table, array, etc.;
2) to develop or choose from a set of genetic operators those that best take into account the features of the search space;
3) to determine the size of the initial population;
4) to develop a technique for using genetic operators;
5)to set the fitness function (the objective function by that variants are selected in the population);
6) to develop a method of selection's variants to a new population;
7) to set the criterion for stopping the evolutionary process.

To minimize the learning error of neural networks based on the genetic algorithm, each variant of the vector of weighting coefficients W is matched with some chromosome, presented in the form of a bit string. In the selection process, a directed search is made for chromosomes that provide the extremum of a given objective function, as the neural network learning error function α is used in the neural network training procedure.

The concept of learning neural network when using a genetic algorithm, in contrast to traditional learning methods, has a different meaning: learning here consists in applying genetic operators to the genotype of the vector W_i, i.e. to the chromosome, and the training sample serves to calculate the learning error α of the neural network with specific values of the weighting coefficients W_i.

Thus, the procedure for optimizing the learning process of neural network using genetic algorithm is also iterative and includes the stages of synthesis of new chromosomes and their selection into the new population.

The scheme of the genetic algorithm of the learning neural network procedure is shown in Fig. 1.

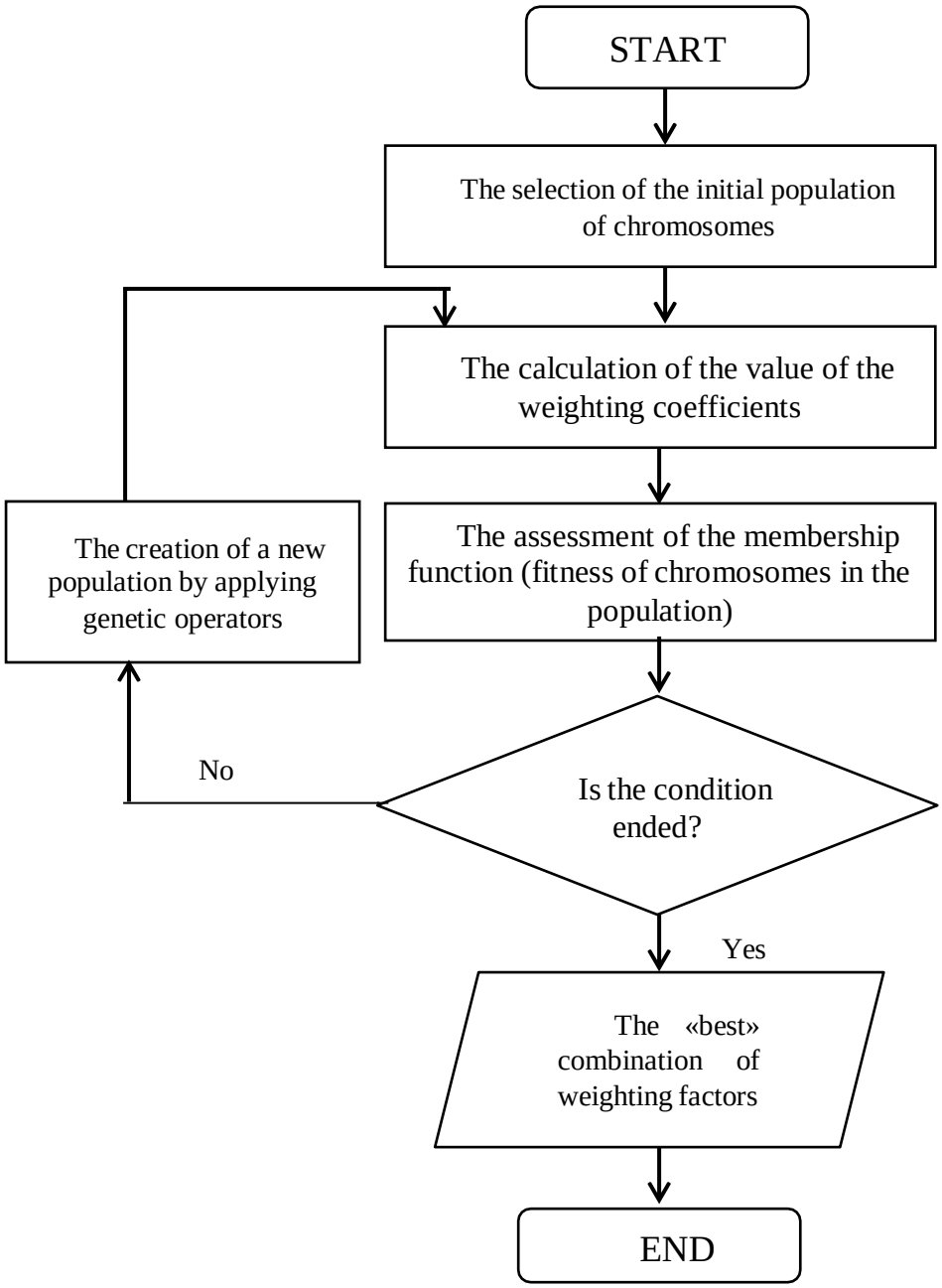

Figure 1: Scheme of the genetic algorithm of the learning neural network procedure

The process continues until an optimal solution or a given number of generations is obtained. At the same time, each subsequent population must be better than the previous one. The solution to this problem is a chromosome with the minimum value of the membership function, that determines the optimal vector of weighting coefficients W_i, while the learning error α becomes less than the given value δ. If the optimal solution isn't found, and the given number of generations is reached, then the learning procedure ends with the selection of an elite chromosome in one or more generations.

Depending on the type of genetic operators and selection schemes used, different genetic algorithms can be constructed, each of them will be effective in terms of convergence speed and the best approach to the extremum when solving real problems.

When solving the problem of forecasting the time series of the temperature of the natural environment, an appropriate neural networks were synthesized, where the input and output values are the temperature values.

As a result of solving the optimization problem by the gradient method of the shortest descent, the best NNs were selected: the radial basis function (errors: training - 2.617 0C, control - 2.617 0C, test - 2.06 0C), the linear with two neurons in the input layer (errors: training - 0.103 0C, control - 0.086 0C, test - 0.097 0C), the linear with three neurons in the input layer (errors: training - 0.103 0C, control - 0.086 0C, test - 0.096 0C), the multilayer perceptron with five neurons in the hidden layer (errors: training - 0.077 0C, control - 0.068 0C, test - 0.074 0C), the multilayer perceptron with two neurons in the hidden layer (errors: training - 0.073 0C, control - 0.065 0C, test - 0.07 0C).

To implement the learning algorithm of the neural network with the help of genetic algorithms, it will use the network with the smallest errors - a multilayer perceptron with two neurons in the hidden layer (Fig. 2).

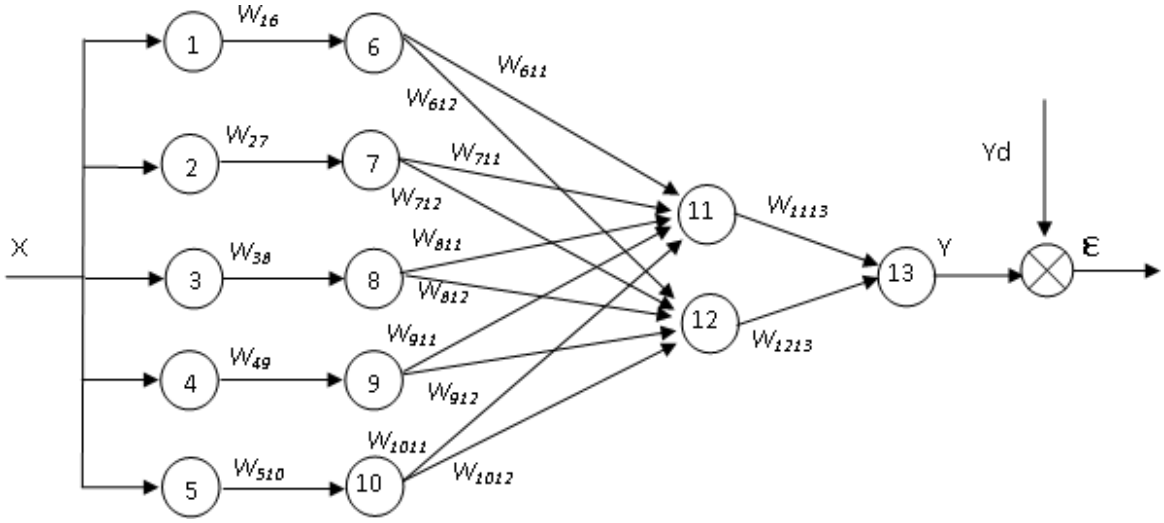

Figure 2: Architecture of NN forecasting of temperature time series: multilayer perceptron with two neurons in the hidden layer (MLP 2)

The parameters of the solved problem are the weighting coefficients W, that is, the problem will have 17 parameters, and the set of these parameters determines the point of the search space, and accordingly, the possible solution.

We assume that the solution search area D is a finite set of solutions, in that each admissible solution $W \in D$ is an n-dimensional vector $\bar{W}= W_1, W_2, \dots, W_n)$, where $n = 17$. Suppose that each component W_i, $i = 1, n$ of the vector $W \in D$ can take values from 0 to 1with a step of 0.004, then the value of the weighting coefficient W_i is encoded using a non-negative integer $x_i \in [0, K_j]$, where $j = 0{,}250$ K_j is the number of possible discrete values of the i-th variable in the search area D [5-7]. To minimize the learning error of the neural network based on the genetic algorithm, we will match each variant of the vector of weighting coefficients with a chromosome, presented in the form of a bit string.

That is, let's match each vector W_i with a vector x_i , for the representation of that in the binary code, it's necessary to determine the maximum number of binary symbols g , that is sufficient to represent any value x_i from the range of its permissible values $x_i \in [0, K_j]$. The value g must meet the requirement $k \leq 2^g$, where $k = 251$ is the number of possible discrete values x_i of the variable, then $g = 8$ (Fig.3).

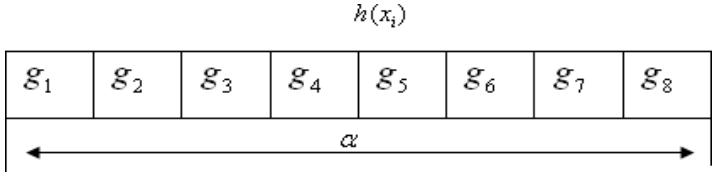

Figure 3: Symbolic representation of fixed variable value x_i in binary code

Where g_m - the binary symbol 0 or 1 ($m = \overline{1, g}$); $h(x_i)$ - the symbolic record of a variable x_i. To present an admissible solution $x \in D$, it's necessary to combine the symbol records of code sequences describing all n components of the vector $\bar{x} = (x_1, x_2, \ldots, x_n)$, where $n = 17$, in the form of a linear sequence of bit strings from binary symbols (Fig. 4).

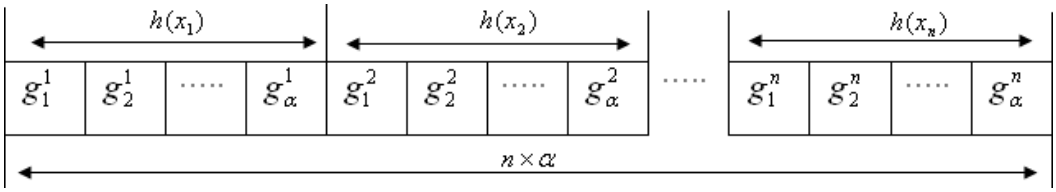

Figure 4: Symbolic notation of the solution vector

Thus, the symbolic model of the solution vector of the given problem $x \in D$, can be represented in the form of a bit string, that is used to specify a set of admissible solutions x_i, belonging to the search domain D.

A binary combination $h(x_i)$ is taken as a gene, that determines the fixed value of the parameter W_i in the binary code. And the smallest indivisible unit amenable to evolution is a person H^1 (r - the number of the person in the population, t - the moment in time of the evolutionary process), characterized by n-genes, each of that is responsible for the corresponding variable (Fig. 5).

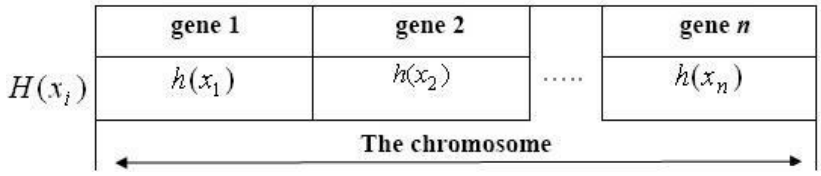

Figure 5: Genotypic chromosome

The chromosome that has specific values of alleles in its loci is called a genotype, that contains all the hereditary genetic information of a person H_r^1. The finite set of all admissible genotypes is the gene pool.

The assessment of the fitness of chromosomes in the population will be determined by calculating the membership function (fitness function) for each chromosome of this population. In our case, this assessment is performed using the membership function, that determines the difference between the calculated and real output value with the same input action and represents the numerical value of the function calculated for an admissible solution to the problem $x \in D - \alpha(H^1) = (Y - Y)^2$, and the smaller the value of the membership function, the better the quality of the chromosome. The fitness function always takes a non-negative value, in addition, to solve the optimization problem, this function needs to be minimized.

A collection of individuals $(H_1^t, \dots, H_r^t)$ forms a population P^t, where p - the size of the population, and $t = 0,1, \dots, T$, where T determines the period of its evolution. The goal of population evolution is to increase the average value of the membership function of the population as a whole:

$$Sit_{CP}(t) = \frac{1}{t} \sum_{1}^{p} Sit_i \quad (3)$$

The effectiveness of the genetic algorithm, the quality of the obtained solution and the evolution as a whole largely depends on the structure and quality of the initial population.

Thus, the optimization procedure using the genetic algorithm is iterative and includes two stages:

- the synthesis of new chromosomes (crossing and mutation);
- the selection of chromosomes in a new population.

The process continues until the optimal solution and the given number of generations are obtained. At the same time, the goal of population evolution (2) is taken into account, that is, each subsequent population should be better than the previous one. The solution of the problem corresponds to the chromosome with the minimum value of the membership function, that determines the optimal vector of weighting coefficients W_i, while the learning error (1) is less than the specified value α_{min}. If the stop of the algorithm cannot be fulfilled according to the condition, then the procedure is completed according to the option and with the selection of the best chromosome in one or more generations.

The optimal weighting coefficients of a multilayer perceptron type neural network with two neurons in the hidden layer (Fig. 1) were determined using a genetic algorithm with the following parameters: the number of chromosomes in the population is 10; the number of populations - from 20 to 100; crossover operator - one-point crossover; percentage of gene mutation - 0.001; the selection is elite. Figure 6 shows the probability of finding extrema of the function from a given number of populations.

The obtained results show that the speed of approaching the extremum is high and increases with the increase in the number of populations [9-13]. It's also possible to draw a conclusion about the low percentage of experiments in that the exact value of the minimum was obtained, depending on the total number of runs of the genetic algorithm, that is, on the number of populations [9-15].

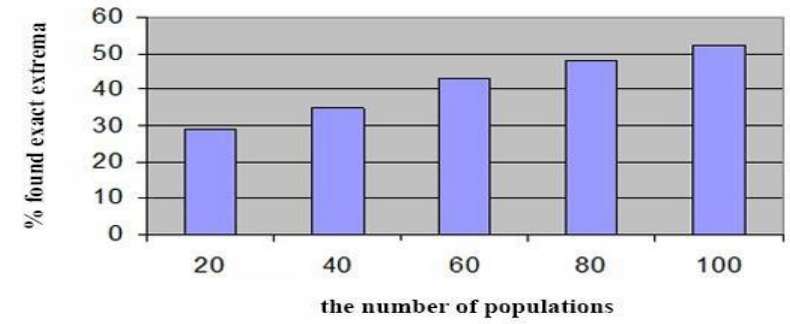

Figure 6: Probability of finding extrema of the function for a given number of populations

Analyzing the projections of time series of temperature (Fig. 7) [9], it should be noted that NN relatively correctly predicted its decrease, increase and stabilization.

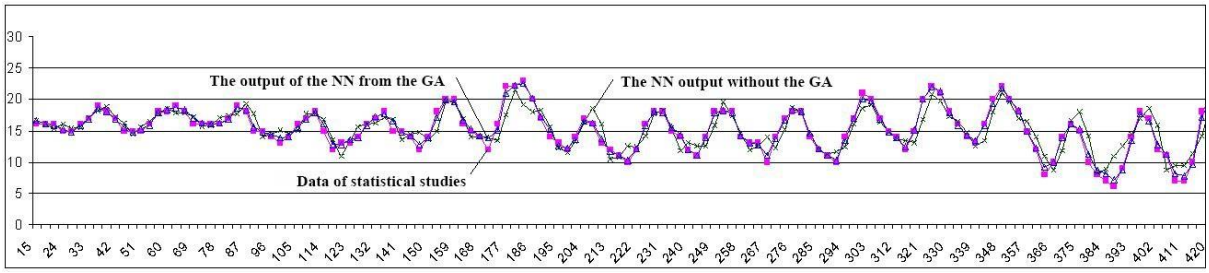

Figure 7: Graphs of deep learning prediction of temperature time series by neural networks

4. Conclusion

1. Experiments on learning neural networks showed that known methods of local and global optimization (gradient, stochastic, Newton, Hessian, etc.) require a significant number of learning steps, that are sensitive to the accuracy of calculations, require a significant number of additional variables. In most cases, they make it possible to find a local rather than a global extremum. That is why the search and development of new methods of learning neural networks is an urgent task.

2. In order to increase the reliability of decisions made on the basis of a neural network, it's necessary to investigate alternative optimization algorithms that allow finding the global extremum. Therefore, there is a need to use such approaches, that wouldn't have the mentioned disadvantages. The genetic optimization algorithms are the most promising in this regard.

3. The scientific and methodical foundations of learning neural networks are developed in the paper. In contrast to traditional learning methods, the concept of learning a neural network using a genetic algorithm is defined, that is the most effective method of optimizing weighting coefficients that minimize the value of the error of the network.

4. The study of the effectiveness of the application of the genetic algorithm to optimize the functioning and deep learning of neural networks in operation established the perspective of such an approach for biotechnical complexes that are exposed to external disturbances (external temperature).

5. References

[1] M. O. Korchemny, V. P. Lysenko, M. V. Chapny, V. M. Shtepa, Neural Networks, Kyiv, Agrar Media Group, 2010.

[2] M. O. Korchemny, V. P. Lysenko, M. V. Chapny, V. M. Shtepa, Neural Networks: Theory and Practice, 4th ed., Kyiv, 2013.

[3] I. Bratko, Algorithms of artificial intelligence in the PROLOG language, Moscow, Williams, 2004.

[4] C. Igel, M. Kreutz, Operator adaptation in evolutionary computation and its application to structure optimization of neural networks, Neurocomputing, No. 55(1-2), 2003, pp. 347-361.

[5] T. Elsken, J. H. Metzen, F. Hutter, Efficient multi-objective neural architecture search via lamarckian evolution, ArXiv preprint arXiv:1804.09081, 2018.

[6] T. Elsken, J. H. Metzen, F. Hutter, The search for neural architecture: an overview. Journal of Machine Learning Research, No. 20(1), 2019.

[7] A. A. Ridkokasha, K. K. Golder, Basics of Artificial Intelligence Systems, Cherkasy, ECHO-PLUS, 2002.

[8] V. V. Kruglov, Artificial neural networks. Theory and practice, Moscow, Hotline – Telecom, 2002.

[9] V. P. Lysenko, N. A. Zayets, V. M. Shtepa, A. O. Dudnyk, Neural Network Forecasting of Time Series of Ambient Temperature, Bioresources and nature conservation 3. 3-4 (2011) 102-108.

[10] N. A. Zayets, V. M. Shtepa, The Use of A Genetic Algorithm For Solving Optimization Problems In Electrical Engineering, Scientific Bulletin of the National University of Bioresources and Nature Management of Ukraine 166. 4 (2011).

[11] N. A. Zayets, S. A. Shvorov, V. M. Shtepa, V. O. Osypa, The Use of Genetic Algorithms to Calculate the Optimal Settings for the Operation of a Robotic Complex, Collection of Scientific Works of the Military Institute of Taras Shevchenko Kyiv National University 38 (2012).

[12] N. A. Pasichnyk, S. A. Shvorov, Y. A. Gunchenko, I. Sharipova, T. M. Tereshchenko, Methodological Bases of Construction of Dispatchers Intensive Training Simulators of AirTraffic Control, in: Proceedings of IEEE 6th International Conferenceon Methods and Systems of Navigation and Motion Control, MSNMC, 2020, pp. 122–125.

[13] S. Shvorov, V. Lysenko, N. Pasichnyk, V. Lukin, A. Martsyfei, The method of determining the amount of yield based on the results of remote sensing obtained using UAV on the example of wheat, in: Proceedings of 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2020, 2020, pp. 245–248.

[14] Y. A. Gunchenko, S. A. Shvorov, N. D. Rudnichenko, V. D. Boyko, Methodical complex of accelerated training for operators of unmanned aerial vehicles, in: Proceedings of IEEE 4th International Conference Methods and Systems of Navigation and Motion Control, MSNMC 2016, 2016, pp. 130–133.

[15] A. N. Voronin, A. G. Yasinsk, S. A. Shvorov. Synthesis of compromise-optimal trajectories of mobile objects in conflict environment, Journal of Automation and Information Sciences 34(2) (2002).