Modification of the genetic method for neuroevolution synthesis of neural network models for medical diagnosis

Serhii Leoshchenko[1[0000-0001-5099-5518]], Andrii Oliinyk[2[0000-0002-6740-6078]], Sergey Subbotin[3[0000-0001-5814-8268]], Nataliia Gorobii[4[0000-0003-2505-928X]] and Vadym Shkarupylo[5[0000-0002-0523-8910]]

[1,2,3,4] Dept. of Software Tools, Zaporizhzhia National Technical University, Zaporizhzhia 69063, Ukraine

[5] Dept. of Computer Systems and Networks, National University of Life and Environmental Sciences of Ukraine, Kyiv 03041, Ukraine

sergleo.zntu@gmail.com, olejnikaa@gmail.com, subbotin@zntu.edu.ua, gorobiy.natalya@gmail.com, shkarupylo.vadym@nubip.edu.ua

Abstract. The main aim of the paper is researching the possibility of application artificial neural networks as the neural network models that can be used in medical diagnostics. One of the most problematic and complex issues of neural network models implementation is the initial stage of synthesis. The article presents a comparison of existing methods of synthesis, as well as a new method. The experiments confirm the effectiveness and expediency of the proposed method.

Keywords: artificial neural networks, synthesis, neuroevolution, genetic method, support vector machine.

1 Introduction

The diagnosis stage plays a crucial role in medicine. Timely accurate diagnosis facilitates the choice of therapy and significantly increases the probability of treatment of the patient. The using of neural networks is one of the ways to improve the efficiency of medical diagnosis [1].

The accuracy of the diagnosis and the speed with which it can be delivered depend on many factors: the patient's condition, the available data on the symptoms and signs of the disease and the results of laboratory tests, the total amount of medical information on the observation of such symptoms in a variety of diseases and, finally, the qualification of the doctor. A major role in this process is played by the human factor, which often leads to errors [1], [2].

Some of the specific medical diagnosis difficulties that need to be considered are listed below [3].

The basis for a reliable diagnosis is a wealth of practical experience, which can be reached only the middle of a doctor's career and is absent at the end of training, of

course. This is especially true for rare or new diseases, where experienced doctors are in the same situation as beginners.

The quality of diagnosis depends on the skill, knowledge and intuition of the doctor.

Emotional problems and fatigue adversely affect the work of the doctor.

Training of specialists is a long and expensive procedure, and therefore in many, even in developed countries, there is a lack of skills.

Medicine is one of the fastest growing and developing fields of science. New results disqualify the old ones, new drugs appear every day. The same applies to the diseases themselves, which take new forms.

These factors necessitate the search for new solutions and tools, for example, the use of artificial neural networks (ANNs) [2].

2 Using the ANN in medical diagnosis

The ANN technologies are designed to solve difficult-to-formalize problems, which, in particular, are reduced to many problems of medicine [4], [5]. This is primarily due to the fact that the researcher is often provided with a large number of heterogeneous factual material for which a mathematical model has not yet been created. In addition, it is necessary to present the results of the analysis in a form, which will be understandable to the specialist. So ANN is a powerful and flexible method of simulating processes and phenomena. Neural networks can be different in structure and form, but they have several common features. A distinctive feature of neural networks is their ability to training on the basis of experimental data of the subject area. With regard to medical subjects, experimental data are presented in the form of a set of initial features or parameters of the object and the diagnosis based on them. Training of ANN is an interactive process in which the neural network finds hidden nonlinear relationships between the initial parameters and the final diagnosis, as well as the optimal combination of weight coefficients of neurons connecting adjacent layers, in which the classification error tends to a minimum [6]. In the training process, the input of the neural network is fed a sequence of initial parameters along with the diagnoses that characterize these parameters. Careful formation of the training sample determines the quality of work, as well as the level of error of the neural network.

A number of difficulties are associated with the use of neural networks in practical problems. One of the main problems of application ANN technologies is a previously unknown degree of complexity of the projected ANN, which will be enough for a reliable diagnosis. This complexity can be unacceptably high and will require more complex network architecture. It is known, for example, that the simplest single-layer neural networks are able to solve only linearly separated problems [7]. This limitation can be overcome by using multilayer neural networks.

The basis of ANNs are neurons with a structure similar to biological analogues. Each neuron can be represented as a microprocessor with several inputs and one output. When neurons are joined together, a structure is formed, which calls a neural network. Vertically aligned neurons form layers: input, hidden and output. The num-

ber of layers determines the complexity and, at the same time, the functionality of the network, which is not fully investigated.

For researchers, the first stage of creating a network is the most difficult task. The following recommendations are given in the literature [8–10].

1. The number of neurons in the hidden layer is determined empirically, but in most cases the rule is used $N_h \leq N_i + N_o$, where N_h is the number of neurons in the hidden layer, N_i in the input and N_o output layers.
2. Increasing the number of inputs and outputs of the network leads to the need to increase the number of neurons in the hidden layer.
3. For the ANNs modeling multistage processes required additional hidden layer, but, on the other hand, the addition of hidden layers may lead to overwriting and the wrong decision at the output of the network.

Based on these recommendations, the number of layers and the number of neurons in the hidden layers is chosen by the researcher, based on his personal experience.

3 Review of the literature

The ANN are attractive from an intuitive point of view, because they are based on a primitive biological model of nervous systems. In this connection, there is an assumption that to improve it may be appropriate to apply another borrowing from nature, for example, evolutionary calculations and, in particular, neuroevolution. Neuroevolution in this paper refers to the automatic modification of neural networks using genetic algorithms. With this methodology, possible variations of neural networks with different topologies are grown, which with each iteration, called generation solve the problem better. Despite genetic programming, as well as evolutionary calculations in general, do not guarantee finding the optimal result, this approach eventually allows us to come to the results applicable to solving practical problems. However, it will take a reasonable amount of time to achieve such results. Thus, the level of complexity of the neural network that arises when it is necessary to create a neural network is significantly reduced, because when it is created, it is only necessary to select the parameter that evaluates the work of the neural network and provide a suitable set of data.

Despite the fact that most of the works devoted to the neuroevolutionary approach offer only a theoretical approach to solving problems of neural network optimization, it is possible to find several promising and noteworthy methods [11–14].

From the early works of noteworthy cellular Frederick Gruau method [15], [16] uses a special grammar for the representation of neural network structures. One individual represented an entire neural network, with each neuron considered as a biological cell, and the growth of the network was determined through the mechanisms of sequential and parallel "division" of neurons - cells. However, this method involves the implementation of a large number of specific operators that provide simulation of cell activity.

The Hierarchical SANE (Symbiotic, Adaptive NeuroEvolution) [17] method uses a different approach. It is consider the development of two independent populations, one of which individuals are separate neurons, and the other contains information about the structures of an artificial neural network. The disadvantages of this method include the fact that the number of hidden neurons and connections is limited.

The ESP method [18] is a development of the sane method. Its main difference is that the network structure is fixed and is given a priori. The population of neurons is divided into subpopulations, in each of which the evolution is independent. Due to parallelization of the solution search, as well as simplification of the problem due to the rejection of the evolution of the artificial neural network structure, ESP works much faster than SANE, sometimes by an order of magnitude, but for the successful operation of the method it is required to choose the appropriate structure of the neural network [19].

One of the most potentially successful attempts to get rid of the disadvantages of direct coding while preserving all its advantages is the method proposed in 2002, called NEAT — Neural Evolution through Augmenting Topologies [15], [20]. Designed by Kenneth Stanley, the NEAT method allows to customize the structure of the network, and without restrictions on its complexity. The solution proposed by the authors is based on the biological concept of homologous genes (alleles), as well as on the existence in nature of the synapsis process — the alignment of homologous genes before the crossover. The technique assumes that two genes (in two different individuals) are homologous if they are the result of the same mutation in the past. In other words, with each structural mutation (gene addition), a new gene is assigned a unique number, which then does not change during evolution. The method uses a number of techniques, such as historical labels and specialization of individuals, to make the process of evolution significantly more efficient [21].

Summing up, it can be noted that the joint use of evolutionary methods and artificial neural networks allows us to solve the problems of configuration and training of artificial neural networks both individually and simultaneously. One of the advantages of this synthesized approach is largely a unified approach to solving a variety of problems of classification, approximation, control and modeling. The use of qualitative evaluation of the functioning of artificial neural networks allows the use of neuroevolutionary methods to solve the problems of the study of adaptive behavior of intelligent agents, the search for game strategies, signal processing. Despite the fact that the number of problems and open questions concerning the development and application of neuroevolutionary methods (coding methods, genetic operators, methods of analysis, etc.) is large, often for the successful solution of the problem with the use of neuroevolutionary method adequate understanding of the problem and neuroevolutionary approach, as evidenced by a large number of interesting and successful works in this direction [15].

4 Materials and methods

In the method, which is proposed to find a solution using a population of neural networks: $P = \{NN_1, NN_2, ..., NN_n\}$, that is, each individual is a separate ANN $Ind_i \rightarrow NN_i$ [19–21]. During initialization population divided into two halves, the genes $g_{Ind_i} = \{g_1, g_2, ..., g_n\}$ of the first half of the individuals is randomly assigned

$g_{Ind_i} = \{g_1 = \text{Rand}, g_2 = \text{Rand}, ..., g_n = \text{Rand}\}$. Genes of the second half of the population are defined as the inversion of genes of the first half $g_{Ind_i} = \overline{\{g_1 = \text{Rand}, g_2 = \text{Rand}, ..., g_n = \text{Rand}\}}$. This allows for a uniform distribution of single and zero bits in the population to minimize the probability of early convergence of the method ($p \to \min$).

After initialization, all individuals have coded networks in their genes with-out hidden neurons (N_h), and all input neurons (N_i) are connected to each output neuron (N_o). That is, at first, all the presented ANNs differ only in the weights of the interneuron connection wi. In the process of evaluation, based on the genetic information of the individual under consideration, a neural network is first built, and then its performance is checked, which determines the fitness function ($f_{fitness}$) of the individual. After evaluation, all individuals are sorted in order of reduced fitness, and a more successful half of the sorted population is allowed to cross, with the best individual immediately moving to the next generation. In the process of reproduction, each individual is crossed with a randomly selected individual from among those selected for crossing. The resulting two descend-ants are added to the new generation $G = P` = \{Ind_1, Ind_2, ..., Ind_n\}$. Once a new generation is formed the mutation operator starts working. However, it is important to note that the selection of the truncation significantly reduces the diversity within the population, leading to an early convergence of the algorithm, so the probability of mutation is chosen to be rather large ($p_{mut} = 15 - 25\%$) [22].

If the best individual in the population does not change for a certain number of generations (by default, it is proposed to set this number at eight), this individual is forcibly removed, and a new best individual is randomly selected from the queue. This makes it possible to realize the exit from the areas of local minima due to the relief of the objective function, as well as a large degree of convergence of individuals in one generation. The general scheme of the method demonstrated at Fig.1.

4.1 Using of genetic operators

It is obvious that the chosen method requires special genetic operators that implement crossover and mutation.

At crossover two parental individuals which produce two descendants are used. Common neurons and connections are inherited by both offspring, and the value of connections in the networks of descendants are formed by a two-point crossover. Elements of ANN, of distinct played out between generations.

An important feature is that neurons with the same indices are considered identical, despite the different number of connections and position in the network, as well as the fact that one of these neurons could have a different index, which changed as a result of correction of indices after mutation. For this purpose, three coefficients were introduced that regulate the size and direction of the network.

The first of them characterizes the degree of connectedness of neurons in the network and is calculated by the formula:

$$f_{con} = \frac{N_c}{2^{FB-1}\left[N_s(N_s-1) - N_i(N_i-1) - (1-FB)N_o(N_o-1)\right]} \quad (1)$$

where N_c is the number of connections in the network, N_i, N_o, N_s are respectively, the number of input, output neurons and the total number of neurons in the network, FB is a variable indicating the permitted occurrence of feedbacks (FB =1) or not (FB =0). It is worth noting that connections from hidden neurons to the output can appear in any case. Thus, the smaller f_{con} the more likely a new relationship will be added as a result of the mutation [23].

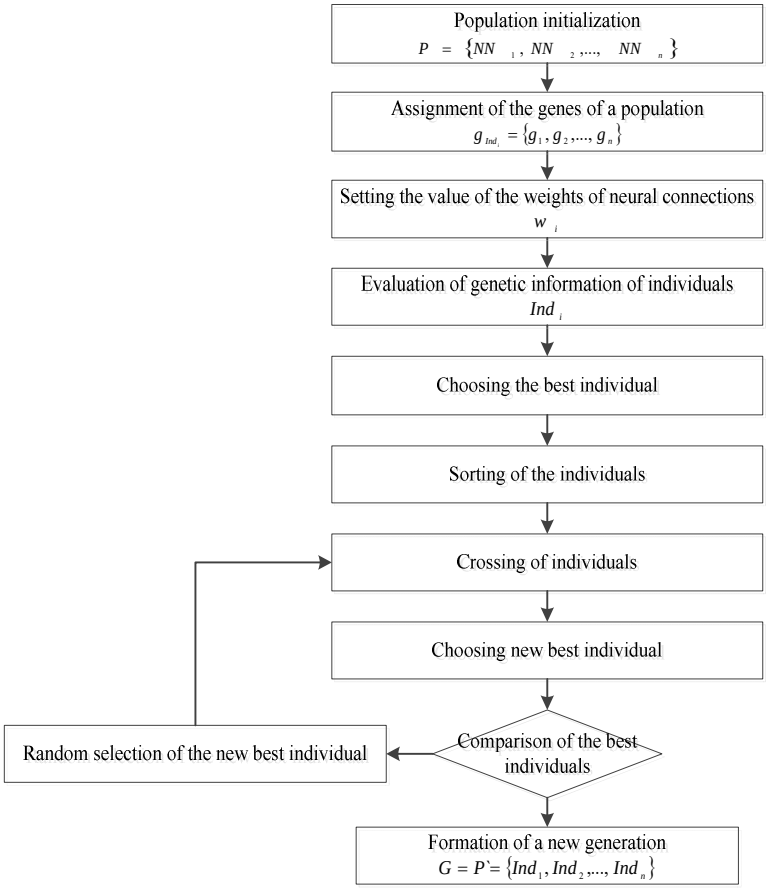

Fig. 1. The general scheme of the method

The use of the second coefficient is based on the assumption that the more elements in the sum of the input and output vectors of the training choice (the greater the total number of input and output neurons), which is probably a more complex network is necessary to solve the problem. The second coefficient is calculated by the formula:

$$f_{top.diff} = \frac{N_i + N_o}{N_s} \tag{2}$$

That is, the more neurons in the network, the less will be $f_{top.diff}$ and the less likely will be selected mutation that adds a new hidden neuron [23].

The third criterion is also based on the assumption that a more complex network should be used to solve more complex problems. However, this criterion characterizes the conditional complexity of the network. This criterion is based on the concept of cyclomatic complexity [24], [25].

$$f_{comp.diff} = \frac{N_i + N_o}{N_s} \tag{3}$$

For any of the described cases, the algorithm uses a ligament $f_{con} \cdot f_{top.diff} \cdot f_{comp.diff}$, because for use it is necessary to take into account the degree of connectivity of already existing neurons.

Thus, using mutations can be pointwise to change the parameters of the structure of the ins.

Chaotic the addition (removal) of neurons and connections can lead to situations where, for example, in a network of many neurons and few connections. It would be more logical to apply different types of mutations depending on the features of the network architecture represented by the mutating individual [26–28].

Removing links gives a side effect: there may be hanging neurons that have no incoming connections, as well as dead-end neurons, that is, without output connections [26], [27], [29]. In cases where the function of neuronal activation is such that at zero weighted sum of inputs its value is not equal to zero, the presence of hanging neurons makes it possible to adjust the neural displacement. It is worth noting that, on the other hand, the removal of links may contribute to the removal of some uninformative and uninformative input features.

4.2 Choosing the mutation type

Consider the dependence of the type of mutation on the values f_{con}, $f_{top.diff}$ and $f_{comp.diff}$. Adaptive mutation mechanism is one of the key features of the proposed method.

The choice of mutation type is determined based on the values of f_{con}, $f_{top.diff}$ and $f_{comp.diff}$. This approach, on the one hand, does not limit the number of hidden neurons from above, on the other hand, it prevents the immeasurable increase of the network, because the addition of each new neuron to the network will be less likely. The mutation of the weight of a random existing bond occurs for all mutating individuals with a probability of 0.5.

Let us consider in more detail how to choose the type of mutation. Fig. 2 shows the block diagram of the selection of the type of mutation. Here RV is a random variable, N_h is the number of hidden neurons in the mutating network.

Conventionally, the entire algorithm can be divided into two branches on the first conditional transition:

– branch increase f_c is carried out for the fulfilment of the conditions of transition;
– branch reduction f_c , performed if the transition condition is not met.

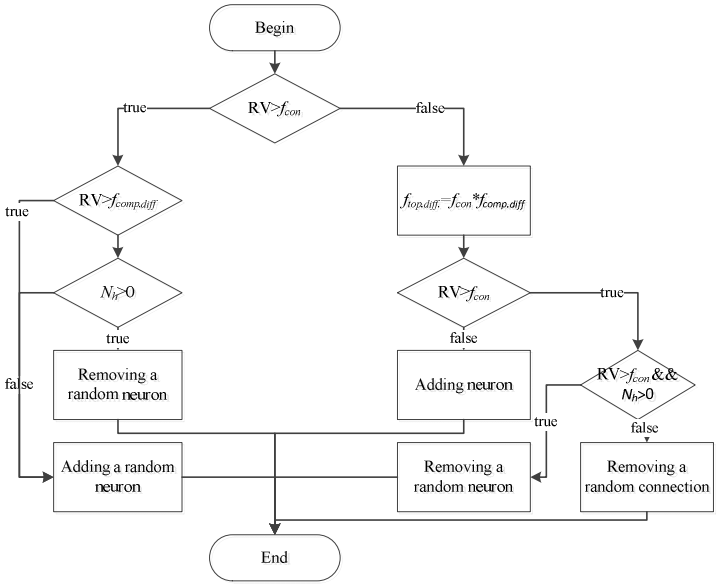

Fig. 2. The diagram of the selection of the type of mutation

Multiplication $f_{con} \cdot f_{comp.diff}$ is necessary in order to change the number of neurons adequately network topology, because the addition (removal) of neurons need information about the feasibility of changes. This information can be obtained indirectly from the value of the characteristic.

4.3 The calculation of the output layer of ANN

On condition using the support vector machine, the optimality criterion for calculating the output weights may not be specified. If the value of the mean square error is replaced by the criterion of the maximum separation of the support vectors, then the optimal linear weights of the output can be estimated using, for example, quadratic programming, as in the traditional method of support vectors, for this it is advisable to use the Evoke operator [30], by the formula:

$$y(t) = w_0 + \sum_{i=1}^{k} \sum_{j=0}^{l_i} w_{i,j} K\left(\phi(t), \phi^i(j)\right), \tag{4}$$

where $\phi(t) \in R^n$ is the output of a recurrent neural network $f(\cdot)$ at a time t; $K(\cdot,\cdot)$ is a predefined kernel function; $w_{i,j}$ is weights corresponding to k training sequences ϕ^i, each length l_i, and are calculated using the support vector machine.

The value of the mean square error is replaced by the criterion of maximum separation of support vectors. In this case, the optimal linear weights can be estimated using quadratic programming, as in the traditional support vector machine.

One of the problems of neuroevolutionary method realization is the algorithm of ANN output calculation with arbitrary topology.

ANN can be represented as a directed planar graph. Based on the fact that the network structure can be any, loops and cycles containing any nodes are allowed in the graph, except for the nodes of the corresponding input neurons. Let denote the set of nodes of the graph by $V = \{v_i \mid i \in [0; N_v - 1]\}$, and a set of arcs through $E = \{e_j \mid j \in [0; N_e - 1]\}$, where N_v and N_e are accordingly, the number of nodes and arcs in the graph, and $N_v = N_s$, and $N_e = N_c$. The arc, which goes from node *k* to node 1 denote by an ordered pair $c_{k,l} = (v_k, v_l)$, the weight of the corresponding link will be denoted by $w_{k,l}$.

Give the index to the nodes of the graph as neurons, that is, the nodes that are the input neurons, called input, have an index out of range $[0; N_l - 1]$. By analogy, the indexes of outgoing nodes belong to the interval $[N_l; N_l + N_o - 1]$, and indexes for hidden nodes will be set in the interval $[N_l + N_o; N_v - 1]$.

Let introduce an additional characteristic for all nodes of the graph equal to the minimum length of the chain to any of the input nodes and denote it ch_i. Let's call ch_i the layer to which the i^{th} node belongs. Thus, all input nodes belong to the 0^{th} layer, not all input nodes that have input arcs from the input belong to the 1st layer, all other nodes with input arcs from nodes of the 1st layer will belong to the layer with index 2, etc .in this case, there may be situations when the node does not have input arcs, we will call it a hanging node with the layer number $ch_i = -1$.

For arcs, we also introduce an additional characteristic $b_{k,l}$ for the arc $c_{k,l}$, which is necessary to determine whether the arc corresponds to forward or reverse. It will be calculated as follows:

$$b_{k,l} = \begin{cases} 1, ch_l - ch_k > 0 \\ -1, ch_l - ch_k \le 0 \end{cases} \tag{5}$$

That is, if the index of the layer of the end node of the arc is greater than the index of the layer of the beginning node, then we will consider such an arc as a straight line, otherwise we will consider the arc as an inverse.

Since each node of the graph represents a neuron, we denote by sum_i the value of the weighted sum of inputs, and through o_i is the value of the output (the value of the activation function of the i^{th} neuron-node). Then, $o_i = f_{fitness}(sum_i)$ where $f_{fitness}$ is the function of neuron activation.

Let's divide the whole process of signal propagation from the input nodes into stages, and during one such stage the signals manage to pass only one arc. The number of the stage is denoted by *s*. For the very first stage *s=1*. For short assumed that all arcs have the same length, and the signals are sewn on them instantly. We denote the feature that the output of node *i* was updated at this stage through a_i, that is, if a_i=1, then the output of the node at stage *s* is calculated, otherwise, if $a_i = 1$ – not.

Let's introduce one more designation $X = \{x_i \mid i \in [0; N_l - 1]\}$ it is vector of input signals. Then the algorithm for calculating the ANN output is as follows:

1. $o_i = x_i$, $a_i = 1$, for all $i \in [0; N_l - 1]$;
2. $o_i = 0$, for all $i \in [N_l; N_s - 1]$;
3. *s=1*;
4. $sum_i = 0$, $a_i = 1$, for all $i \in [N_l; N_s - 1]$;
5. if $s = 1$, than go to the step number 7;
6. calcultion of the feedback network. For all input feedbacks $c_{j,k}$ node v_k, where $k \in [N_l; N_s - 1]$: $sum_k = sum_k + o_j$, if $ch_j < s$;
7. if $a_i = 0$, than $fn(i)$ for all $i \in [N_l; N_s - 1]$;
8. if the stop criterion is not met, than *s=s+1* and go to the step number 4.

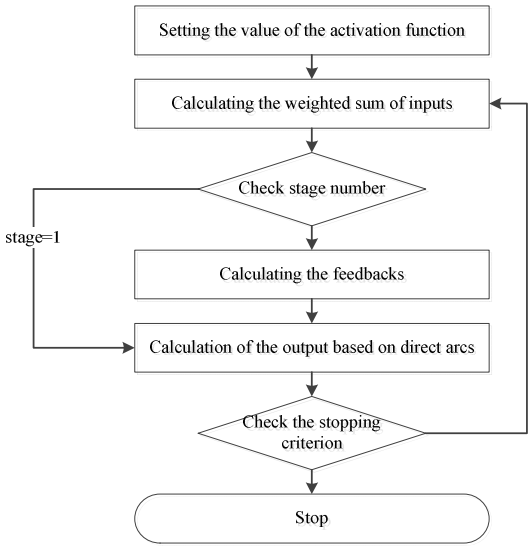

Fig. 3. The general scheme of the calculation of the output layer of ANN

Here $fn(i)$ is a recursive function that calculates the output of the 1^{st} node taking into account all straight arcs. Works on the following algorithm:

1. if $ch_i < 0$, than go to the step number 3;
2. for all input arcs $c_{k,l}$ node v_i : if $a_k = 1$, than $sum_i = sum_i + o_k$, else $fn(k)$;
3. $o_i = f(sum_i)$;
4. exit.

The stopping criterion of the ANN output calculation algorithm can be one of the following:

- stabilization of values at the output of ANN;
- s exceeds the set value.

It is more reliable to calculate the output until the values at the output of ANN do not change, but for the case when the network contains cycles and/or loops, its output may never become stable. Therefore, the required additional stopping criteria limiting the maximum number of stages of calculation of network output. For networks with no feedback (FB =0) in many cases, allow the $\max(ch_i)+1$ phases.

5 Experiments

During testing, the main task is to track the speed of the proposed method, quality and stability. Since synthesized ANN can be further used as diagnostic models for medical diagnosis, testing should be carried out on the relevant test data. Also, testing will be carried out in 2 stages: the first stage will consist in the synthesis of ANN only with the help of modified genetic algorithm, and the second – in additional processing of the initial layer by the support vector machine. This strategy will allow to know more clearly how useful the support vector machine is.

Data for testing were taken from the open repository – UC Irvine Machine Learning Repository. Data sample was used: Breast Cancer Coimbra Data Set [31]. Clinical features were observed or measured for 64 patients with breast cancer and 52 healthy controls. There are 10 predictors, all quantitative, and a binary dependent variable, indicating the presence or absence of breast cancer. The predictors are anthropometric data and parameters which can be gathered in routine blood analysis. Prediction models based on these predictors, if accurate, can potentially be used as a biomarker of breast cancer. Table 1 shows the main characteristics of the data sample.

Table 1. Main characteristics of the Breast Cancer Coimbra Data Set

Criterion	Characteristic	Criterion	Characteristic
Data Set Characteristics	Multivariate	Number of Instances	116
Attribute Characteristics	Integer	Number of Attributes	10

During the evaluation of the test results we will pay attention to the following criteria:

- the spent time, s;
- average error of final network (E);
- standard deviation (SD).

The relative error value in this case will be calculated as the ratio of the classification error to the total sample size (number of instances).

$$E = \frac{error_{class}}{Number_{sampl}} \cdot 100\%, \tag{6}$$

where E is relative error; $error_{class}$ is classification error; $Number_{sampl}$ the number of instances in the sample.

Standard deviation gives an idea about how one or the other, the ANN accurately predicts the user's rating, since the estimation is calculated the difference between the result of the work of ANN's and known result. It is also important to know that this indicator can be calculated only with a sufficient amount of observations. Otherwise, the calculation of the SD will be uninformative and its use will not lead to improvement of the results of the ANN.

$$SD = \sqrt{\frac{1}{Number_{sampl}} \sum_{i=1}^{Number_{sampl}} \left(x_i - \bar{x}\right)^2}, \tag{7}$$

where SD is standard deviation, x_i is i^{th} element of the set, $Number_{sampl}$ the number of instances in the sample, $\bar{x}$ is the mean value of these observations [32].

6 The results analysis

Table 2 shows the results of testing the modified genetic algorithm in comparison with the NEAT and ESP methods.

Table 2. Results of testing

	Time, s	E	SD
NEAT	468.013	4.40%	3.70
ESP	9389.55	3.07%	2.99
Modified GA	631.373	2.96%	3.05

As the table shows, the modified GA according to the time of fulfillment is ahead of ESP in time, however, inferior to the NEAT. However, it should be noted that in the analysis of error values, the proposed method is significantly ahead of existing methods.

Let's repeat testing, but now with additional use of the support vector machine. The results are shown in Table 3.

Table 3. Results of testing

	Time, s	E	SD
NEAT	468.013	4.40%	3.70
ESP	9389.55	3.07%	2.99
Modified GA (with support vector machines)	5326.326	2.36%	2.17

As can be seen from the table, the modified GA using the support vector machine is inferior to the opponents in terms of execution time. However, on indicators of minimum, maximum errors and average errors of the output ANN significantly better than their competitors. Therefore, we can conclude that the use of the support vector machine really significantly improves the results of the synthesis.

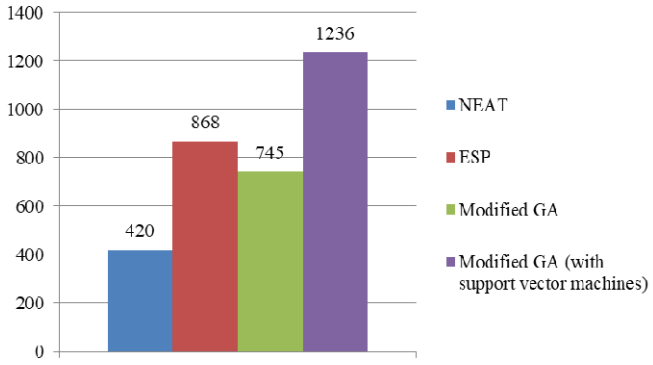

Fig. 4. The distribution of iterations in experiments

As you can see from the diagram, the modified genetic method was more iterative than the existing methods, but the time spent on iteration was less. That is, it can be concluded that iterations are not complex and for their reduction it is possible to resort to parallelization, which will significantly speed up the work even when using the support vector machine [33–35].

7 Conclusion

The problem of finding the optimal method of synthesis of ANN requires a comprehensive approach. Existing methods of ANNs training are well tested, but they have a number of nuances and disadvantages. The paper proposes a mechanism for the use a modified genetic algorithm for its subsequent application in the synthesis of ANNs. Based on the analysis of the experimental results, it can be argued about the good

work of the proposed method. However, to reduce iterativity and improve accuracy, it should be continued to work towards parallelization of calculations.

Acknowledgment

The work was performed as part of the project "Methods and means of decision-making for data processing in intellectual recognition systems" (number of state registration 0117U003920) of Zaporizhzhia National Technical University.

References

1. Committee on Diagnostic Error in Health Care, Board on Health Care Services, Institute of Medicine, The National Academies of Sciences, Engineering, and Medicine: Improving Diagnosis in Health Care. In: Balogh, E., Miller, B., Ball, J. National Academies Press, Washington, USA (2015).
2. Lugovskaya, A. M.: Artificial neural networks in medical diagnostics [Iskusstvennyie neyronnyie seti v meditsinskoy diagnostike]. Computer systems and networks: proceedings of the 54th scientific conference of postgraduates, undergraduates and students 2018, vol. 1, pp. 182 – 183. BGUIR, Minsk (2018).
3. Goold, S.D., Lipkin, M.Jr.: The doctor-patient relationship: challenges, opportunities, and strategies. Journal of General Internal Medicine (1), 23–33 (1999).
4. Kolpakova, T., Oliinyk, A., Lovkin, V.: Improved method of group decision making in expert systems based on competitive agents selection. IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), Institute of Electrical and Electronics Engineers, pp. 939–943, Kyiv (2017). doi: 10.1109/UKRCON.2017.8100388
5. Stepanenko, O., Oliinyk, A., Deineha, L., Zaiko, T.: Development of the method for decomposition of superpositions of unknown pulsed signals using the secondorder adaptive spectral analysis. EasternEuropean Journal of Enterprise Technologies, vol. 92, issue 2/9, pp. 48–54, (2018). doi: 10.15587/1729-4061.2018.126578
6. Callan, R.: The Essence of Neural Networks (The Essence of Computing Series). Prentice Hall, Europe (1999).
7. Yasnitskiy, L.N.: Introduction to artificial intelligence [Vvedenie v iskusstvennyiy intellekt]. 3th edn. Academy, Moscow (2010).
8. Bondarenko, I.B., Gatchin, Yu.A., Geranichev, V.N.: Cintez optimalnyih iskusstvennyih neyronnyih setey s pomoschyu modifitsirovannogo geneticheskogo algoritma. Nauchno-tehnicheskiy vestnik informatsionnyih tehnologiy, mehaniki i optiki, vol. 2 (78), 51–55 pp. (2012).
9. Lukichev, D.V., Usoltsev, A.A.: Sintez optimalnoy strukturyi neyrosetevyih ustroystv, 97–102 (2005).
10. Van Tuc, N.: Approximation contexts in addressing graph data structures. University of Wollongong Thesis Collection, 30–55 (2015).
11. Oliinyk, A., Skrupsky, S., Subbotin, S., Korobiichuk, I.: Parallel Method of Production Rules Extraction Based on Computational Intelligence. Aut. Control Comp. Sci. (2017) 51 pp. 215-223. DOI: 10.3103/S0146411617040058
12. Oliinyk, A., Subbotin, S., Lovkin, V., Leoshchenko, S., Zaiko, T.: Development of the indicator set of the features informativeness estimation for recognition and diagnostic model synthesis. 14th International Conference on Advanced Trends in Radioelectronics, Tele-

communications and Computer Engineering (TCSET 2018), pp. 903–908, (2018). doi: 10.1109/TCSET.2018.8336342.
13. Oliinyk, A., Subbotin, S., Lovkin, V., Leoshchenko, S., Zaiko, T.: Feature Selection Based on Parallel Stochastic Computing," 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), pp. 347-351. Lviv (2018). doi: 10.1109/STC-CSIT.2018.8526729
14. Oliinyk A., Fedorchenko I., Stepanenko A., Rud M., Goncharenko D.: Evolutionary method for solving the traveling salesman problem. Problems of Infocommunications. Science and Technology : 5th International Scientific-Practical Conference PICST2018, Kharkiv, 9–12 October 2018, Kharkiv, Kharkiv National University of Radioelectronics, pp. 331-339 (2018). doi: 10.1109/INFOCOMMST.2018.8632033.
15. Tsoy, Y.R.: Evolutionary Algorithms Design: State of the Art and Future Perspectives. Proceedings of IEEE East-West Design and Test Workshop (EWDTW'06), Sochi, Russia, pp. 375-379 (2006).
16. Gruau, F.: Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. In Proceedings of the International Workshop on Combination of Genetic Algorithms and Neural Networks (COGANN-92). Los Alamos, CA: IEEE Computer Society Press, pp. 55-74 (1992).
17. Moriarty, D., David, R., Miikkulainen, R.: Hierarchical evolution of neural networks. Evolutionary Computation Proceedings, pp. 428–433 (1998). doi: 10.1109/ICEC.1998.699793.
18. Greer, B., Hakonen, H., Lahdelma, R., Miikkulainen, R.: Numerical optimization with neuroevolution. Evolutionary Computation CEC '02(1), pp. 396–401 (2002). doi: 10.1109/CEC.2002.1006267.
19. Enforced Subpopulations (ESP) neuroevolution algorithm for balancing inverted double pendulum, http://blog.otoro.net/2015/03/10/esp-algorithm-for-double-pendulum.
20. Stanley, K.O., Miikkulainen, R.: Evolving Neural Networks through Augmenting Topologies. The MIT Press Journals, vol. 10, num. 2, pp. 99-127 (2002).
21. Whiteson, S., Whiteson, D.: Stochastic optimization for collision selection in high energy physics. Proceedings of the 19th national conference on Innovative applications of artificial intelligence, IAAI'07, vol. 2, pp. 1819-1825 (2007).
22. Hochreiter, S., Schmidhuber, J.: Long Short–Term Memory. Neural Computation, vol. 9. issue 8, pp. 1735–1780 (1997).
23. Tsoy Yu.R.: Development of a genetic algorithm for setting up an artificial neural network [Razrabotka geneticheskogo algoritma nastroyki iskusstvennoy neyronnoy seti]. Tomskiy politehnicheskiy universitet (2004).
24. Papadimitriou, F.: Mathematical modelling of land use and landscape complexity with ultrametric topology. Journal of Land Use Science vol. 8(2), pp. 1-21 (2011).
25. Papadimitriou, F.: Artificial Intelligence in modelling the complexity of Mediterranean landscape transformations. Computers and Electronics in Agriculture, pp. 87-96 (2012).
26. Leoshchenko, S., Oliinyk, A., Subbotin, S., Gorobii, N., Zaiko, T.: Synthesis of artificial neural networks using a modified genetic algorithm. Proceedings of the 1st International Workshop on Informatics & Data-Driven Medicine (IDDM 2018), pp. 1-13 (2018). dblp key: conf/iddm/PerovaBSKR18
27. Yarymbash, D., Yarymbash, S., Kotsur, M., Divchuk, T.: Analysis of inrush currents of the unloaded transformer using the circuitfield modelling methods. Eastern-European Journal of Enterprise Technologies, vol. 3, № 5 (93), pp. 6-11 (2018). doi: 10.15587/1729-4061.2018.134248
28. Yarymbash, D., Yarymbash, S., Kotsur, M., Divchuk, T.: Enhancing the effectiveness of calculation of parameters for short circuit of threephase transformers using field simulation

methods. Eastern-European Journal of Enterprise Technologies, vol. 4, № 5 (94), pp. 22-28 (2018). doi: 10.15587/1729-4061.2018.140236.
29. Shkarupylo, V., Skrupsky, S., Oliinyk, A., Kolpakova T.: Development of stratified approach to software defined networks simulation. EasternEuropean Journal of Enterprise Technologies, vol. 89, issue 5/9, pp. 67–73 (2017). doi: 10.15587/1729-4061.2017.110142
30. Schmidhuber, J., Wierstra, D., Gagliolo, M., Gomez, F.: Training Recurrent Networks by Evolino. Neural computation. vol. 19(3), 757–779 pp. (2007). doi: 10.1162/neco.2007.19.3.757.
31. Breast Cancer Coimbra Data Set, https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra
32. Bland, J.M., Altman, D.G.: Statistics notes: measurement error. BMJ. 312 (7047): 1654 (1996). doi:10.1136/bmj.312.7047.1654
33. Oliinyk, A., Leoshchenko, S., Lovkin, V., Subbotin, S., Zaiko, T.: Parallel data reduction method for complex technical objects and processes. 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018), 526–532 pp. (2018). doi: 10.1109/DESSERT.2018.8409184 IEEE Catalog number: CFP18P47-ART 978-1-5386-5903-8.
34. Leoshchenko, S., Oliinyk, A., Subbotin, S., Zaiko, T.: Methods of semantic proximity extraction between the lexical units in infocommunication systems. 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T 2017), 7–13 pp. (2017). doi: 10.1109/INFOCOMMST.2017.8246137.
35. Alsayaydeh, J.A., Shkarupylo, V., Hamid, M.S., Skrupsky, S., Oliinyk, A.: Stratified Model of the Internet of Things Infrastructure, Journal of Engineering and Applied Science, vol. 13, issue 20, pp. 8634-8638, (2018). doi: 10.3923/jeasci.2018.8634.8638.