An Integrated ANN-GA Approach to Data Classification

Stanislav Alkhasov, Alexander Tselykh, Alexey Tselykh
Department of Information Security Systems
Southern Federal University
Taganrog, Russia
{alkhasov,ant,tselykh}@sfedu.ru

***Abstract*— In this paper, we present an advanced approach to data classification based on the integration of artificial neural networks (ANNs) and genetic algorithms (GAs). We modify neural network architecture in a two-stage process. During the first stage, GA finds a suboptimal neural network architecture: number of nodes, training algorithm, learning rate, etc. Then, the fitting of weight coefficients and bias is carried out in order to minimize GA fitness function. In final section of the paper, we compare the results of the conventional and the proposed approaches.**

Keywords—classification; artificial neural networks; genetic algorithms; ANN-GA.

I. INTRODUCTION

Artificial neural networks (ANNs) are one of the ways to tackle the problems of data classification. ANNs are an important tool of data mining. They are used in many applied studies in the field of churn prediction, credit scoring, document classification, etc. [1].

Gradient methods generally used for ANN training can lead to the penetration of the functional of training quality in local minima. Lately GAs are often used to increase the quality of ANN training. GAs perform tasks of artificial evolution of main features of neural network structure in order to increase classification accuracy [2].

In [3], two possible variants of ANN architecture optimization are described. In the first case, the chromosomes have the same length equal to the number of ANN layers, and each gene characterizes the number of nodes in the corresponding layer. If the layer is absent then in the corresponding locus the zero gene is prescribed. Another approach assumes that chromosomes have unequal length, and the first locus contains the gene responsible for the number of ANN layers, and following genes contain data about the number of nodes. In this case, the null values are not entered into the chromosome. So the chromosomes have unequal length. In order to perform the operation of crossing-over in the second case the chromosome with shorter length is modified zero genes occupying random loci in the range between the third and the penultimate loci.

In [4], the enhanced modification of the above approach is proposed involving the study of the influence of not only the number of layers and nodes but also other heterogeneous characteristics of ANN (transfer function, learning rate, etc.) to achieve a suboptimum of the quality functional [5].

It is possible when in the population of the chromosomes that consist of the ANN weights and bias the identical values of gene series that characterize individual neu-rons are in different loci. The crossing-over of such chromosomes leads to the permu-tation problem [6] (or competing conventions problem [7]). In [8], for the problem solution the author recommends usage of the inversion operator. However, in this case the chromosomes must either be homogeneous or consist only of the elements of the single array, for example, the array of weights in the ANN hidden layer.

In [9], the algorithm ESP is represented. The authors set the architecture of the neural network a priori. The neurons were considered as chromosomes here. The neuron population was splitted into the subpopulations. For each subpopulation, the evolution is independent. It allows parallelize the search of the solution [10].

In [11], the type of "+/–" strategy is proposed for the change of the population size. The population size rises if the fitness of the best chromosome decreases or not changes (the progress is absent). In the case of evolutional improvement one should reduce the population size. It allows, among other things, to decrease the volume of necessary computing resources during the GA execution.

The solution of various classification problems is not always obvious. So it is actual to apply of genetic algorithms for intelligent selection of suboptimal classificator parameters. A feature of this work is the combined use of two interconnected different genetic algorithms to build the neural network of a particular type and then to adjust its weight coefficients and biases.

II. GA-OPTIMIZATION OF THE ANN STRUCTURE

The GA-training of the ANN can include two stages. At the first stage, the GA finds a suboptimal architecture of the ANN. The second step determines suboptimal values of weight coefficients and bias. The first stage is considered a preparatory step of training when the subsequent task of determining of weights and bias is simplified [12].

In this study, we used the following approach: chromosomes were built as vectors containing number of nodes in the hidden layer, learning rate, type of transfer function, type of training algorithm, and other some parameters. Due to the heterogeneity of these parameters the use of the inversion operator is not possible in the general case.

The number of additional genes includes the vector which characterizing the occurrence of one or the other variable in the training array. Zeros and ones were distributed randomly to loci with the only remark that the number of zeros must not exceed half of the original number of variables. The introduction of this type of genes into the chromosomes was carried out to identity suboptimal set of variables [13].

After determining the ANN architecture, the run of network training is performed. At the end of this stage the augmented column vector $n \times 1$ of weights and bias is formed as well as the vector of mean square error (MSE). From series of the vectors (chromosomes) the initial population is created which then will evaluate via the per-formance of the GA [14].

At this stage, it is permissible to apply in the GA the inversion operator within homogeneous loci belonging to one of four possible groups: weights $\mathbf{w}_1$ and bias $\mathbf{b}_1$ of the hidden layer and weights $\mathbf{w}_2$ and bias $\mathbf{b}_2$ of the output layer.

III. Research Methodology

In the research, the classification problem was tackled for real estate objects with series of features (total square, living square, floor no., number of floors, price, etc.). The samples were separated into two classes corresponding to primary and secondary real estate market.

Because of the large number of features (variables) the convergence during GA-optimization is achieved over a long period of time. This spends significant computing resources but the classification quality increases [15].

We consider a combined approach involving preliminary determination of weights and bias by using one of the methods of backpropagation. The obtained weights and bias are grouped in the chromosomes, made up the initial population.

Here the fitness function of the GA is the functional ϕ of the ANN training quality [16].

$$\phi = \frac{1}{\Upsilon}\sum_{\upsilon=1}^{\Upsilon} \mathrm{H}_{\upsilon}^2 ,$$

where Υ is the number of the elements in the output array, H_{υ} is the classification error of each ANN output for each sample.

Investigation of the effectiveness of the model is performed via the below presented program of actions.

for $i = \overline{1, I}$

$$r_i = \mathrm{rand} ,$$

where rand is random number generator.

$$\left[\dot{\mathbf{X}}_i, \ddot{\mathbf{X}}_i\right] = \mathrm{separate}\left(\mathbf{X}, r_i\right),$$

where separate is the function of the separation of the initial array to two new arrays in accordance with the r_i, $\mathbf{X}$ is the initial input array, $\dot{\mathbf{X}}_i$ is the input array for the fitting of ANN parameters, $\ddot{\mathbf{X}}_i$ is additional test array of input data. Such notations of these functions are used to simplify the described algorithms here and below in the paper.

Below it is similarly for target values **T**.

$$\left[\dot{\mathbf{T}}_i, \ddot{\mathbf{T}}_i\right] = \mathrm{separate}\left(\mathbf{T}, r_i\right).$$

for $j = \overline{1, J}$

$$\left[h_{i,j}, f_{i,j}\right] = \mathrm{ga}' F\left(\hbar\right),$$

where $h_{i,j}$ is the chromosome, $f_{i,j}$ is a value of the fitness function for the chromosome $h_{i,j}$, ga' is the genetic algorithm for the first part of this GA-model, $F\left(\hbar\right)$ is the fitness function.

$$\mathrm{net}'_{i,j} = \mathrm{configure}\left(\mathrm{MLP}, h_{i,j}\right),$$

where $\mathrm{net}'_{i,j}$ is the artificial neural network, configure is the configuration function for multi-layer perceptron (MLP) with neural network architecture determined by the chromosome $h_{i,j}$.

for $k = \overline{1, K}$

$$\mathrm{net}'_{i,j,k} = \mathrm{train}\left(\mathrm{net}'_{i,j}, \dot{\mathbf{X}}_i, \dot{\mathbf{T}}_i\right),$$

where train is the ANN training function [17].

$$e'_{i,j,k} = \frac{1}{\tilde{N}'}\sum_{\tilde{n}'=1}^{\tilde{N}'}\left(\mathrm{sim}\left(\mathrm{net}'_{i,j,k}, \ddot{\mathbf{X}}_i\right) - \ddot{\mathbf{T}}_i\right)^2_{\tilde{n}'},$$

where $e'_{i,j,k}$ is MSE of the classification in result of simulation sim of ANN using the additional test array as input data, $\tilde{N}' = 2\ddot{N}'$ is the number of samples in the array whose size is the product of the number of ANN output layers and the number of samples in the set.

end

$$\breve{e}'_{i,j} = \frac{1}{K}\sum_{k=1}^{K} e'_{i,j,k} ,$$

where $\breve{e}'_{i,j}$ is the average score of MSEs $e'_{i,j,k}$.

$$\breve{s}'_{i,j} = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(e'_{i,j,k} - \breve{e}'_{i,j}\right)^2} ,$$

where $\breve{s}'_{i,j}$ is the standard deviation of MSEs $e'_{i,j,k}$.

end

$$h_i^* = \arg\min_{j=1}^{J} h_{i,j}\left(\breve{e}'_{i,j}\right),$$

where h_i^* is j-th vector of the array **h**. It corresponds to minimum value in the array $\breve{\mathbf{e}}'$.

end

$$h^{**} = \arg\min_{i=1}^{I} h_i^*\left(\breve{e}'_{i,j}\Big|_{h_i^* = h_{i,j}}\right),$$

where h^{**} is i-th vector of the array $\mathbf{h}^*$. It corresponds to the minimum value in the array $\breve{\mathbf{e}}'$ and is the optimal chromosome which contains data about suboptimal architecture of the ANN.

After the choice of the optimal chromosome the second stage of the described program is begun. Suboptimal values of weight coefficients and bias are determined for the ANN with the configuration $\mathbf{h}^{**}$.

for $\lambda = \overline{1, \Lambda}$

$$r_\lambda = \text{rand},$$

$$\left[\dot{\mathbf{X}}_\lambda, \ddot{\mathbf{X}}_\lambda\right] = \text{separate}\left(\mathbf{X}, r_\lambda\right),$$

$$\left[\dot{\mathbf{T}}_\lambda, \ddot{\mathbf{T}}_\lambda\right] = \text{separate}\left(\mathbf{T}, r_\lambda\right);$$

for $\mu = \overline{1, M}$

$$\text{net}''_{\lambda,\mu} = \text{configure}\left(\text{MLP}, h^{**}\right),$$

if $\lambda \neq 1$,

where the inequality $\lambda \neq 1$ means that for the first run $\lambda = 1$ the weights and bias are determined with a standard way in accordance with the algorithm of Widrow – Nguen [18].

then

$$\text{net}''_{\lambda,\mu} = \text{setwb}\left(\text{net}''_{\lambda,\mu-1}, G^*_{\lambda,\mu-1}\right),$$

where setwb is the function of the enter into ANN modified values $G^*_{\lambda,\mu-1}$ of weights and bias.

end

$$\text{net}''_{\lambda,\mu} = \text{train}\left(\text{net}''_{\lambda,\mu}, \dot{\mathbf{X}}_\lambda, \dot{\mathbf{T}}_\lambda\right),$$

$$\varepsilon''_{\lambda,\mu} = \frac{1}{\tilde{N}''}\sum_{\tilde{n}''=1}^{\tilde{N}''}\left(\text{sim}\left(\text{net}''_{\lambda,\mu}, \ddot{\mathbf{X}}_\lambda\right) - \ddot{\mathbf{T}}_\lambda\right)^2_{\tilde{n}''},$$

where $\varepsilon''_{\lambda,\mu}$ is MSE of the classification in result of simulation sim of ANN using the additional test array as input data, $\tilde{N}'' = 2\ddot{N}''$ is the number of samples in the array whose size is the product of the number of ANN output layers and the number of samples in the set.

$$\gamma_{\lambda,\mu} = \text{formwb}\left(\text{net}''_{\lambda,\mu}\right),$$

where formwb is the function of the output from ANN of current values of weight coefficients and bias.

$$\mathbf{P}_{\lambda,\mu} = \left[\boldsymbol{\gamma}_{\lambda.} \quad \mathbf{G}_{\lambda.}\right]$$

where **P** is the initial population with the number of "gradient" and "genetic" chromosomes equal 2μ. The chromosome which did not win the competition is preserved in the gene pool of the initial population primarily as a material for crossing-over.

for $\psi = \overline{1, \Psi}$

$$\left[g_{\lambda,\mu,\psi}, \varphi_{\lambda,\mu,\psi}\right] = \text{ga}''\left(\Phi\left(\gamma_{\lambda,\mu}\right), \mathbf{P}_{\lambda.}\right),$$

where ga'' is the genetic algorithm in which the initial population **P** is used.

$$\overline{\text{net}''}_{\lambda,\mu,\psi} = \text{setwb}\left(\text{net}''_{\lambda,\mu}, g_{\lambda,\mu,\psi}\right)$$

$$e''_{\lambda,\mu,\psi} = \frac{1}{\tilde{N}''}\sum_{\tilde{n}''=1}^{\tilde{N}''}\left(\text{sim}\left(\overline{\text{net}''}_{\lambda,\mu,\psi}, \ddot{\mathbf{X}}_\lambda\right) - \ddot{\mathbf{T}}_\lambda\right)^2_{\tilde{n}''},$$

where $e''_{\lambda,\mu,\psi}$ is MSE of the classification which is performed by the GA-modified neural network.

end

$$\breve{e}''_{\lambda,\mu} = \arg\min_{\psi=1}^{\Psi} e''_{\lambda,\mu,\psi}\left(\varphi_{\lambda,\mu,\psi}\right)$$

$$G_{\lambda,\mu} = \arg\min_{\psi=1}^{\Psi} g_{\lambda,\mu,\psi}\left(\varphi_{\lambda,\mu,\psi}\right)$$

if $\varepsilon''_{\lambda,\mu} > \breve{e}''_{\lambda,\mu}$,

where the above inequality expresses the competition between "gradient" and "genetic" chromosomes.

then $G^*_{\lambda,\mu} = G_{\lambda,\mu}$

else $G^*_{\lambda,\mu} = \gamma_{\lambda,\mu}$

end

end

Then we present formulae for the calculation of the average score and the standard deviation of the above obtained MSEs.

$$\bar{\varepsilon}''_\lambda = \frac{1}{M}\sum_{\mu=1}^{M}\varepsilon''_{\lambda,\mu},$$

$$\bar{\sigma}''_\lambda = \sqrt{\frac{1}{M-1}\sum_{\mu=1}^{M}\left(\varepsilon''_{\lambda,\mu} - \bar{\varepsilon}''_\lambda\right)^2}$$

$$\bar{e}''_\lambda = \frac{1}{M}\sum_{\mu=1}^{M}\breve{e}''_{\lambda,\mu},$$

$$\bar{s}''_\lambda = \sqrt{\frac{1}{M-1}\sum_{\mu=1}^{M}\left(\breve{e}''_{\lambda,\mu} - \bar{e}''_\lambda\right)^2}$$

end

$$Æ'' = \frac{1}{\Lambda}\sum_{\lambda=1}^{\Lambda}\bar{\varepsilon}''_\lambda,$$

$$Z'' = \sqrt{\frac{1}{\Lambda-1}\sum_{\lambda=1}^{\Lambda}\left(\bar{\varepsilon}''_\lambda - Æ''\right)^2}$$

$$E'' = \frac{1}{\Lambda}\sum_{\lambda=1}^{\Lambda}\bar{e}''_\lambda,$$

$$S'' = \sqrt{\frac{1}{\Lambda-1}\sum_{\lambda=1}^{\Lambda}\left(\bar{e}''_\lambda - E''\right)^2}$$

In order to evaluate the efficiency of use of the GA the obtained results are compared with the reference model in which the stages of forming of the suboptimal architecture and the weights and bias fitting are absent.

In the reference model the network architecture is selected on the basis of the relation which is a consequence of the theorem of Arnold – Kolmogorov – Hecht-Nielsen [19].

$$\frac{mN}{1+\log_2 N} \leq L_w \leq m\left(\frac{N}{m}+1\right)(n+m+1)+m,$$

$$L = \frac{L_w}{n+m}$$

It is also known the following relation [20].

$$\frac{N}{10} - n - m \leq L \leq \frac{N}{2} - n - m$$

where *n* is the number of the ANN inputs, *m* is the number of the ANN outputs, *N* is the number of samples in the training set [21].

From both relations, it follows that the choice of the optimal number of nodes in the hidden layer depends on the number of training samples. These relations accept a wide range of possible values in the case of big values of the number of training parameters. For the reference model the number was chosen which close to the lower threshold of the range due to the limited computing resources. In the studied GA-model the alleles of the population chromosomes that contain information about the number of the hidden layer nodes had values within the given range [22].

IV. Experiment

The experiment was conducted in accordance with the methodology presented in the previous section. Initially we determined the efficiency of a conventional neural network model 13 – 40 – 2 based on the use of the algorithm of gradient descent as the training algorithm [23]. The training pair $\left[\dot{\mathbf{X}} \quad \dot{\mathbf{T}}\right]$ and the additional test pair $\left[\ddot{\mathbf{X}} \quad \ddot{\mathbf{T}}\right]$ were formed some number of times in a ratio $\frac{2}{3}:\frac{1}{3}$ to obtain the average score of the training error. The first pair was used for the training and, in turn, divided into training, validation, and test pairs in a ratio 0.70:0.15:0.15, respectively. The second pair was applied for the final efficiency test of the model after the end of the training. Additional sets were used to assess the behavior of the model in the case of new data input. Another characteristic of the repeated training results is the variance that allows to estimate the scattering of the results. This technique of the model training is close to the procedure of cross-validation [24].

In order to quantify the model we also use the coefficient ξ, which shows in how many times the degree of the true recognition of one of the classes (here it is more rare class "1" corresponding to the primary real estate market) is more than the random guessing.

$$\xi = \frac{\tau_1 / \nu_1}{\nu_1 / N} = \frac{\tau_1 N}{\nu_1^2},$$

where τ_1 is the number of the true recognitions of the class "1",

ν_1 is the number of the samples in the target set belonging to the class "1",

N is the total number of all samples.

After the forming of the indicative results the further research was carried out using GA. First one needed to find the ANN suboptimal architecture. The splitting of input data was performed in the same way as in the previous model. In the first part of the GA-model the output is the suboptimal chromosome $\mathbf{h}^{**}$ with genes which configure the ANN in the second part of this GA-model [25].

As a result of performing the first part of the GA-model, the following suboptimal set of ANN parameters is obtained:

- the set of input features: 12 out of 13;
- the number of hidden layer nodes: 46;
- learning rate: 0.1416;
- transfer function: logistic function $y(x) = \frac{1}{1+e^{-x}}$ (the choice among two variants: logistic function and hyperbolic tangent $y(x) = \tanh x$);
- training method: Levenberg – Marquardt algorithm /LMA/ (the choice among 13 variants: algorithm of gradient descent, LMA, RProp algorithm, etc.).

After the forming of the GA-optimized architecture of the ANN the second part of the GA-model is run. It performs for the identification of the suboptimal set of weight coefficients and bias.

The obtained results of the performing of the classification via the conventional neural network approach and the GA-modified approach are summarized in Table I. Confusion matrix is used for the description of the model. It is the matrix

$$\begin{pmatrix} \tau_1 & \chi_2 \\ \chi_1 & \tau_2 \end{pmatrix},$$

where τ_1 and τ_2 is the number of the true recognitions of the class "1" and class "2", respectively,

χ_1 and χ_2 is the number of the false recognitions of the class "1" and class "2", respectively.

TABLE I. Results of the Classification by Means of Conventional and Modified Approaches

Approach	Conventional	GA-modified
The average score of MSE	0.15	0.08
Variance	10^{-3}	$2.4 \cdot 10^{-4}$
Confusion matrix, %	$\begin{pmatrix} 11.95 & 6.65 \\ 9.32 & 72.08 \end{pmatrix}$	$\begin{pmatrix} 17.57 & 8.11 \\ 1.35 & 72.97 \end{pmatrix}$
Coefficient ξ	2.65	4.91
Approximate time of the performing, s	$3 \cdot 10^3$	$7 \cdot 10^4$

V. Conclusions

In this study, we identified the increase of classification accuracy when using GA-modified neural network approach. Based on the principle of evolution, GA reveals the best combination of the ANN's main properties that can vary not only across different tasks, but also in case of changes in data parameters. Thus, given approach allows automated classification operations. Among the shortcomings, it requires an increased volume of computing resources. Therefore, it is recommended to apply cloud computing resources for higher efficiency of the above methodology.

Acknowledgment

The reported study was supported by RFBR, research project No. 15-01-09139 A.

REFERENCES

[1] S.S. Alkhasov, A.N. Tselykh, and A.A. Tselykh "Application of cluster analysis for the assessment of the share of fraud victims among bank card holders," 8th International Conference on Security of Information and Networks, pp. 103-106. New York: ACM, 2015.

[2] Z.Waszczyszyn (Ed.) "Neural Networks in the Analysis and Design of Structures. CISM Courses and Lectures No. 404,", Wien – New York: Springer, 1999.

[3] V.I. Bozhich, O.B. Lebedev, and Yu.L. Shnitser "Razrabotka geneticheskogo algoritma obucheniya neyronnykh setey," (in Russian) Izvestiya TSURE, vol. 22, pp. 170-174, 2001.

[4] E.A. Shumkov and I.K. Chistik "Ispolzovanie geneticheskikh algoritmov dlya obucheniya neyronnykh setey," (in Russian) Politematicheskiy setevoy electronnyy nauchnyy zhurnal KubSAU. vol. 91, pp. 1-9, 2013, http://ej.kubagro.ru/2013/07/pdf/78.pdf.

[5] T.V. Panchenko "Geneticheskie algoritmy," (in Russian). Astrakhan: AstrakhanSU, 2007.

[6] N.J. Radcliffe "Genetic Neural Networks on MIMD Computers," Ph.D. thesis, 1990.

[7] K.O. Stanley and R. Miikkulainen "Evolving Neural Networks through Augmenting Topologies," Evolutionary Computation. vol. 10, pp. 99-127, 2002.

[8] I.R. Ayupov "Parametricheskiy metod obucheniya neyronnoy seti pri reshenii zadach prognozirovaniya," (in Russian) Ph.D. thesis, 2015.

[9] F. Gomez and R.Miikkulainen "Incremental Evolution of Complex General Behavior," Adaptive Behavior, vol. 5, pp. 317-342, 1997.

[10] Yu.R. Tsoy "Neyroevolyutsionnyy algoritm i programmnye sredstva dlya obrabotki izobrazheniy," (in Russian) Ph.D. thesis, 2007.

[11] Yu.R. Tsoy and V.G. Spitsyn "Issledovanie geneticheskogo algoritma s dinamicheski izmenyaemym razmerom populatsii," (in Russian) 5th International Conference on Intelligent Systems and Intelligent CAD systems (IEEE AIS'05/CAD–2005), vol. 1, pp. 241-246. Moscow: FIZMATLIT, 2005.

[12] D. Rutkowska, M. Piliński, and L. Rutkowski "Neyronnye seti, geneticheskie algoritmy i nechetkie sistemy," (in Russian) Moscow: Goryachaya liniya – Telekom, 2006.

[13] L.A. Gladkov, V.V. Kureychik, and V.M. Kureychik (Ed.) "Geneticheskie algoritmy," (in Russian) Moscow: FIZMATLIT, 2006.

[14] A.L. Lyakhov and S.P. Aleshin "Iskusstvennaya neyronnaya set' kak izmeritel'nyy instrument adekvatnosti modeli s adaptivnym klassom toch'nosti," (in Russian) Matematychni mashyny i systemy, vol. 2, pp. 61-66, 2010.

[15] A.A. Barsegyan, M.S. Kupriyanov, V.V. Stepanenko, and I.I. Kholod "Tekhnologii analiza dannykh: Data Mining, Visual Mining, Text Mining, OLAP," (in Russian) St. Petersburg: BKhV-Peterburg, 2008.

[16] V.S. Medvedev and V.G. Potemkin (Ed.) "Neyronnye seti. MATLAB 6," (in Russian) Moscow: DIALOG-MIFI, 2002.

[17] D. Banks, L. House, F.R. McMorris, P. Arabie, and W. Gaul (eds) "Classification, Clustering and Data Mining Applications," Heidelberg: Springer, 2004.

[18] E.V. Burnaev and P.D. Erofeev "Vliyanie initsializatsii parametrov na vremya obucheniya i tochnost' nelineynoy regressionnoy modeli," (in Russian) Informatsionnye protsessy, vol. 15, pp. 279-297, 2015.

[19] Z.I. Sichinava "Neyrosetevye algoritmy analiza povedeniya respondentov," (in Russian) Ph.D. thesis, 2014.

[20] V.V. Kruglov, M.I. Dli, and R.Yu. Golunov "Nechetkaya logika i iskusstvennye neyronnye seti," (in Russian) Moscow: FIZMATLIT, 2001.

[21] D. Stathakis "How many hidden layers and nodes?" International Journal of Remote Sensing. vol. 30, pp. 2133-2147, 2009, http://www.academia.edu/711697/How_many_hidden_layers_and_nodes.

[22] A.J. Thomas, M. Petridis, S.D. Walters, S.M. Gheytassi, and R.E. Morgan "On Prediction the Optimal Number of Hidden Nodes," 2015 International Conference on Computational Science and Computational Intelligence, pp. 565-570, 2015.

[23] G.E. Yakhyaeva "Osnovy teorii neyronnykh setey," (in Russian), http://www.intuit.ru/studies/courses/88/88/info.

[24] G.-B. Huang "Learning capability and storage capacity of two-hidden-layer feedforward networks," IEEE Transactions on Neural Networks. vol. 14, pp. 274-281, 2003.

[25] "Geneticheskie algoritmy v MATLAB," (in Russian), 2011, http://habrahabr.ru/post/111417/.