Indian Journal of Science and Technology, Vol 9(29), DOI: 10.17485/ijst/2016/v9i29/99467, August 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

The Multi-Objective Optimization of Complex Objects Neural Network Models

Vadim Sergeevich Tynchenko*, Valeriya Valerievna Tynchenko, Vladimir Viktorovich Bukhtoyarov, Sergei Vasilyevich Tynchenko and Eduard Arkadievich Petrovskyi

Siberian Federal University, Krasnoyarsk, Russia; vadimond@mail.ru, 051301@mail.ru, vladber@list.ru, 051311@mail.ru, petrovsky_quality@mail.ru

Abstract

Background/Objectives: The study considers the modeling technique that applies artificial neural networks analyzing their types and functional principles. **Methods:** A comparative analysis of the existing methods of structural and parametric synthesis of artificial neural networks has been carried out; the practicability of applying evolutionary approach to solve this problem has been justified. **Findings:** The multi-objective optimization of the structure of a neural network model has been formalized, given its computational complexity. The genetic algorithm has been adjusted to solve the problems of unconditional optimization of the parameters of the neural network and of selecting its effective structure in multi-objective setting. The results of solving the practical problem prove that the application of the suggested approach can help alleviate the computational complexity of the obtained structures of artificial neural networks. **Applications/Improvements:** The results of the study make it possible for a decision maker to select neural network model among multiple options, given the required precision and the available computational resources.

Keywords: Artificial Intelligence, Modeling, Multi-Objective Optimization, Neural Network

1. Introduction

Artificial neural networks have been successfully applied for fulfilling various scientific and technical tasks, such as object recognition automation process, complex object adaptive control, functional approximation, forecasting, expert system creation, content addressed memory arrangement, etc. However, this approach cannot always be effectively employed in practice by wide circle of experts, as there are no properly formalized procedures that would comprehensively encompass the whole process of building neural networks.

The tools of artificial neural networks are founded on modeling the characteristic features of the structure and functions of biological brain cortex. Artificial Neural Network (ANN) represents an aggregate of interrelated Formal Neurons (FN), each of them representing a simplified model of a biological neuron, a fundamental structural and functional unit of the nervous system.

ANN classification can be based upon different attributes, for instance:

- type of communication topology existing between the neurons of the network;
- signal propagation characteristics or signal type;
- type of neuron structure;
- nature of neutron state transition with time, etc.

From the perspective of topology, the following types of ANN and their combinations can be distinguished[1]:

- fully connected;
- multilayer;
- weakly connected (with local connections);
- unstructured (with multiple connections).

Based on signal propagation nature, neuron networks are divided into[2]:

*Author for correspondence

- monotone (special case of multilayer networks with additional preconditions for connections and neurons);
- direct propagation networks;
- feedback networks (the outputs of the network are fed to the inputs).

Based on signal type, neuron networks can be subdivided as follows[3]:

- binary (binary signal);
- analog (continuous signal).

Based on the type of neuron structures, one can distinguish homogenous and heterogeneous ANN.

Based on the nature of neutron state transition in time, there are:

- synchronous ANN;
- asynchronous ANN.

In the course of constructing particular neural network models the abovementioned characteristics can be realized in different ways with the purposes of describing various properties of biological systems. Up to now, great variety of ANNs has been developed and investigated; their construction methods and application areas have been described in many studies[2,4-10]: multilayer perceptrons, the Hopfield networks, the Hemming networks, self-organizing maps (the Kohonen networks), radial basic function networks, counter-propagation networks, cognitron (authored by K. Fukushima), neocognitron, BAM – bidirectional associative memory (the Kosko neural network), ART-networks, networks of adaptive resonance theory (suggested by Carpenter and Grossberg).

This study considers the most general case of ANN topology, where each neuron can be connected with any other neuron in the network. The signal is propagated in one direction only, namely, from inputs to outputs.

2. Concept Headings

ANN function is to transform the input vector into output vector, which is preset by weight factors of the network. Activation functions of all neurons in the network are usually fixed, and the weight could change representing the parameters of ANN.

The process of creating neural network model consists of two staged:

1. Selecting ANN structure.
2. Setting parameters (training) of ANN with the preset structure.

2.1 Methods of Setting ANN Parameters

Adjusting ANN with some certain preset structure means looking for such a set of weight factors that would be best for fulfilling the set tasks in the process of running the obtained neural network model.

The following paradigms of ANN training are distinguished: "with a teacher", "without a teacher" (self-learning) and mixed training[11].

Training "with a teacher" implies using a training set containing learning examples, each of them representing a training couple of vector signals that are fed to neurons of the input layer, and the corresponding vector signals at the output of the neural network (target vector).

The aggregate of the training couples is a learning sample. In the process of training, the inputs of the neural network are randomly fed with input vectors from the learning sample, and the weights of the connections are selected to minimize the total deviation occurring between the real and the required outputs across all learning pairs, that is to minimize the neural network training error.

Training algorithms "without a teacher" include the Hebb method, the Kohonen algorithm, the competition method,[3] the Adaline and Madaline algorithm,[12] linear discriminant analysis,[9] Sammon's projection,[7] principle component analysis.[5] With such an approach to ANN training, there is a predetermined set of input vectors only, and for each of these vectors, in the process of neural network adjustment, the output vector is selected that is supposed to be the best in some certain sense and that is determined by the training algorithm.

If the mixed training algorithm is applied, then the weight factors of one group of neurons are adjusted by means of training "with a teacher", and those of another group are adjusted "without a teacher". The most successful model applying the mixed method of training is a network with linear preference.[5]

In case of ANN training "with a teacher", the task of educating the neural network represents the task of multi-objective optimization that consists of searching

for optimum set of the weights of the neural network that would minimize the target function of error. There are two large groups of algorithms that solve this problem, namely, gradient-based and stochastic algorithms.

Gradient algorithms for network training are found on calculating partial differential coefficient of the error function of the network parameters. This group include, for example, such algorithms as the gradient-based algorithm (the steepest descend algorithm) and the conjugate gradient algorithm. The most efficient gradient-based algorithm is the error back-propagation algorithm.[1] However, this algorithm is applicable only to the special case of ANN training, to fully connected networks with layered signal propagation. Such disadvantages of the gradient-based methods, as local convergence, dependency on the start point location, and dependency on the selected incremental length limit the possibilities of applying these methods to fulfill the task of creating a neural network model.

In stochastic algorithms, the search for the minimum of the error function is performed randomly (the simulated annealing methodology, the Monte-Carlo method, the random direction search).[13] A peculiar feature of the stochastic optimization algorithms is that their application calls for multiple calculations to be performed in the process of building a neural network. Among the algorithms of this group, the genetic adaptation methods gained widespread currency.[1,14]

2.2 Selecting ANN Structure

One of the problems of practical application of neural network modeling, faced by practical researchers, is represented by the lack of effective procedure that would be sufficient to computerize ANN structure selection completely.

The description of the structure of multi-connected neural network includes setting such parameters as total number of neurons, the activation functions in each neuron, and either presence or absence of connections between certain neurons. Often, the structure of the neural network is selected, based on the specific features of the task under consideration and based on the recommendation of an expert who is sufficiently experienced in operating such systems.

To solve the task of selecting the efficient ANN structure, there are two groups of algorithms[6]:

1. Destructive algorithms;
2. Constructive algorithms.

Applying destructive algorithms (abridgement algorithms) implies that, initially, a network is set with the size greater than required for solving the problem. Upon training the network with the redundant structure, this structure is then corrected by deleting the least significant neutrons and connections as long as the increase in the error of the network adjustment remains negligible. The correction procedure is finished when the following potential changes in ANN structures would entail considerable increase in the training error. There are different approaches to the procedure of ANN structure correction. The penalty function method or the projection method is usually employed for multilayer networks.2 For the unstructured networks the contrasting method is applied effectively.[6]

The disadvantages of the abridgement algorithms could include the following:

- no formalized procedure to predetermine the number of redundant neurons and connections in ANN to perform a certain set task;
- it is not clear, what is the reason for failure in the network training: was it the redundancy, or, on the contrary, the insufficient complexity of its initially selected structure;
- from the very beginning, neural network possesses redundant structure, which makes the process of its training slower.

Applying constructive algorithms implies gradual growth of some certain, initially small, ANN structure. The neural network with the preset structure undergoes training, and, if the training is successful, the process is accomplished with obtaining some option of the network configuration; otherwise, extra neurons or connections are added. Thereat, the skills that the network acquired before increasing the number of neurons and connections are preserved. In the course of realizing constructive algorithms several different methods are applied for adding new neurons and connections to the existing ANN structure: random addition, addition to minimize training error, evolutionary structural growth methods. Also, there are methods for setting parameters in the new neurons that are added to the network: weight factors are selected randomly or are determined by disintegrating one of the old neurons.[2]

Applying constructive algorithms helps eliminate the disadvantages intrinsic to destructive algorithms. However, in cases of neural networks with interlayer and incomplete connections as well as in cases with different types of neurons, it is quite difficult to find the way to increase ANN architecture with every step; thereat, the number of possible options grows by several times at each of the subsequent step of the algorithm.

Both constructive and destructive algorithms for selecting ANN structure represent local procedures of searching for structures in space, and this search can be stopped at the nearest, more or less successful structure that could be far from the optimum one. This is a considerable disadvantage of this group of the algorithms.

Selecting efficient ANN structure is similar to the task of adjusting the neural network parameters, and it represents a multi-objective optimization problem with the data structures that differ from the vector real variables and that feature the algorithmically set functions (ANN training procedures). To solve these optimization problems, genetic algorithms (GA) have been applied successfully.[15]

The advantages of applying genetic algorithms for the purposes of structural and parametric synthesis of ANN are given below:

- - they are suitable for optimizing the unstructured neural networks;
- - they ensure global review of the solution space and help avoid the local minimums;
- - they can be applied to fulfilling the tasks where it is either too difficult or too expensive to obtain the data on the gradients;
- - conceivably, they possess the property of massive parallelism in processing, insofar as the species in the population operate independently, which affords effective parallel implementation.

Notwithstanding the abovementioned advantages of the genetic algorithms, it is not possible to be sure that the decision obtained through such an algorithm will make, at the least, a local extremum. The efficiency of the genetic algorithm depends on its parameter settings considerably.[16-21]

For the purposes of computerization of structural and parametric synthesis of the neural network model with random connections between the neurons, complex multi-objective optimization problems will have to be solved to select the efficient structure of the neural network and to adjust its weight factors. To solve such types of optimization problems, genetic algorithms proved to be a very good tool that does not need data on the properties of the optimized function and affords global search in solution space.

2.3 Multi-Objective Setting of Optimization Problem for Synthesis of Structure of Multi-connected Neural Network

In the course of solving the optimization problem of synthesizing the effective structure of the neural network, the root-mean-square error of ANN training that is supposed to be minimized, is usually used as a criterion of evaluating the quality of this neural network.

To reduce the time spent on computations for the neural network model and to ensure the acceptable precision of training simultaneously, it seems practicable that another criterion of optimization should be added that would help minimizing the computational complexity of the structure of the multi-connected neural network.

This judgment leads to the multi-criteria setting of the optimization problem in formulating the structure of neural network model.

The formalized representation of the abovementioned setting will be put as follows:

$$\begin{cases} E(C,W,\overline{af}) \to \min\limits_{C,W,\overline{af}}, \\ CD(C,\overline{af}) \to \min\limits_{C,\overline{af}}, \end{cases}$$

Where $E(.)$ – total root-mean-square error of training;

$CD(.)$ – computational complexity of the neural network structure;

C – matrix of connections, dimension $N_{nrns} \times N_{nrns}$;

W – matrix of the weight factors of connections, dimension $N_{nrns} \times N_{nrns}$;

$\overline{af}$ – vector, setting the activation functions of neurons, dimension N_{nrns};

N_{nrns} – number of neurons.

To calculate total root-mean-square error of training of neural network, apply the formula as follows:

$$E(C,W,\overline{af}) = \frac{1}{m}\sum_{j=1}^{m}\sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(OUT_k^j(C,W,\overline{af}) - y_k^j\right)^2}, \quad (1)$$

Where $OUT_k^j(C,W,\overline{af})$ – real value of the signal at k-output of the neural network when the input of the

neural network is fed with j-image, $(C, W, \overline{af})$ – description of the neural network structure;

y_k^j – ideal value of output signal at k-neuron;

n – number of neurons in output layer;

m – number of learning examples.

The time spent on calculations for the neural network model with the preset structure should be basically determined by the total time spent on processing all connections between neurons and the total time spent on calculating the activation functions at each neuron. Hence, for the purposes of evaluating the computational complexity of the structure of the multi-connected neural network, the following formula could be suggested:

$$cd(C,\overline{af},P)=\sum_{i=1}^{N_{conn}(C)} T_i^{conn}(C,P)+\sum_{i=1}^{N_{nrns}} T_i^{act}(\overline{af},P), \quad (2)$$

where $N_{conn}(.)$ – total number of connections;

$T_i^{conn}(.)$ – time for processing i-connection;

$T_i^{act}(.)$ – time for calculating the activation function at i-neuron;

N_{nrns} – total number of neurons;

P – productivity of the computing system.

Determine the time spent on processing one connection as the time for adding its weight factor to the sum of the weight factors of the connections at the input of a certain neuron, that have been processed earlier:

$$T_i^{conn}(C,P)=T(\sum_{j=1}^{n+1} w_{ij})-T(\sum_{j=1}^{n} w_{ij}),$$

where $T(x)$ – time for calculating x;

w_{ij} – i-input of j-neuron;

$n=0,1,2,\ldots$ – number of connections processed by now.

Assume that the time spent on processing one connection is equal for all connections. Then formula (2) will be represented as follows:

$$cd(C,\overline{af},P)=N_{conn}(C)\cdot T^{conn}(P)+\sum_{i=1}^{N_{nrns}} T_i^{act}(\overline{af},P), \quad (3)$$

where $T^{conn}(P)$ – time for processing one connection.

Move to the time-independent non-dimensional values used for evaluating the computational complexity of the structure of the neural network, dividing, for the purpose, both parts of equation (3) by $T^{conn}(P)$:

$$\frac{cd(C,\overline{af},P)}{T^{conn}(P)}=N_{conn}(C)+\sum_{i=1}^{N_{nrns}}\frac{T_i^{act}(\overline{af},P)}{T^{conn}(P)}.$$

Write down: $CD(C,\overline{af})=\dfrac{cd(C,\overline{af},P)}{T^{conn}(P)}$.

Introduce the quotient that describes the relative computational complexity of the activation function of i-neuron and that does not depend on hardware:

$$K_i(\overline{af})=\frac{T_i^{act}(\overline{af},P)}{T^{conn}(P)}.$$

Than the criterion for evaluating the computational complexity of the neural network will be finally represented as follows:

$$CD(C,\overline{af})=N_{conn}(C)+\sum_{i=1}^{N_{nrns}} K_i(\overline{af}). \quad (4)$$

Given the abovementioned equations for calculating total root-mean-square error of training (1) and the computational complexity of the structure (4) of the neural network model, the multi-objective setting of optimization problem for synthesizing the structure of multi-connected neural network will be represented as follows:

$$\begin{cases} \dfrac{1}{m}\sum_{j=1}^{m}\sqrt{\dfrac{1}{n}\sum_{k=1}^{n}\left(OUT_k^j(C,W,\overline{af})-y_k^j\right)^2}\rightarrow\min\limits_{C,W,af} \\ N_{conn}(C)+\sum_{i=1}^{N_{nrns}} K_i(\overline{af})\rightarrow\min\limits_{C,\overline{af}}. \end{cases} \quad (5)$$

2.4 GA Adaptation for ANN Training

In the course of ANN training applying the evolutionary method, the following basic stages should be followed[1,10]:

- selecting the relevant coding system for the weights of connections.
- running the process of evolution based on the genetic algorithm.

Each chromosome in the population contains the coded set of the weight factors of the neural network. The genetic algorithm applied to the population implements the typical cycle of evolution including four steps as follows:

Step 1. Decoding the species of current generation (restoring the set of the weights) and forming the

neural network with the preset architecture and with the relevant weight factors of the connections between neurons.

Step 2. Calculating total root-mean-square error that occurs between the actual and the preset values at all outputs of the network when its inputs are fed with training images. The specified error characterizes the adaptability (fitness) of the species that is represented by the constructed neural network.

Step 3. Reproduction of species in line with the chosen selection method.

Step 4. Applying such genetic operators as crossing (breeding) and mutation to obtain new generation.

The function of total root-mean-square error of training can be used for the fitness of i -individual:

$$f_i = \frac{1}{1 + GenError_i},$$

$$GenError_i = \frac{\sum_{j=1}^{m-1} Error_j}{m},$$

$$Error_j = \sqrt{\frac{\sum_{k=1}^{n} \left(OUT_k - y_k \right)^2}{n}},$$

Where OUT_k – real signal at the output of k -neuron of the output layer of the neural network when its inputs are fed with j -image;

y_k – ideal value of output signal of this neuron.

2.5 Adapting GA to Select ANN Structure

At the first stage of evolutionary engineering of the neural network architecture, a decision has to be made on the relevant form of its description. First, the general conceptual structure of data representation (matrices, graphs) should be selected. Second, it has to be decided which information characterizing the architecture of the network should be used and which coding scheme should be adopted. There is a scheme of direct coding (strong specification scheme) and a scheme of indirect coding (weak specification scheme). When strong specification scheme is applied, the complete information on the architecture of the neural network is directly coded as binary sequences, that is each connection and each unit (neuron) is directly specified with certain number of bits. When weak specification scheme is applied, then only the most important parameters or properties of the architecture are coded: number of units (neurons), number of connections and the type of the activation function at the neuron.[1,22]

Consider in more detail the direct coding scheme that is used in this study. Matrix C, dimension $n \times n$, $C = [c_{ij}]_{n \times n}$ represents the connections of the neural network that consists of n units (neurons), thereat, value c_{ij} predetermines either presence or absence of the connection between i-neuron and j-neuron. If c_{ij}=1, then the connection exists, if c_{ij}=0, then there is no connection. With such an approach, the binary sequence (chromosome), reflecting the connections of the neural network, is represented as a combination of lines (columns) of matrix C.

This study considers unidirectional connections, which makes it possible to account only for those elements of matrix C that predetermine the connections of this unit (neuron) with the units next to it.

Connection matrix has zero values that can be omitted in the course of coding the chromosome, namely:

- first columns of the connection matrix, corresponding to the input neurons, always contain zeros, insofar as no neurons can be connected with the neurons of the input layer;
- last lines of the matrix, corresponding to output neurons, always contain zeros, insofar as the neurons of the output layer cannot be connected with any neurons;
- the elements of the connection matrix located below the main diagonal are always equal to zero, because only unidirectional connections are accounted for in this study;
- on the main diagonal of the matrix, there are always zeroes, insofar as the neurons cannot be connected with each other.

Besides, the chromosome should contain information on the type of the activation function at each neuron (binary counting number of this function in the set of all activation functions used for solving the problem). This information can be added, for example, in the end of the bit sequence of the relevant neuron. At the input layer, the activation function for each neuron is always represented as $F\_activ\_inp(x) = x$, and therefore, it is not taken into account in the course of coding.

Second stage of evolutionary engineering of the neural network architecture consists of the steps as follows:

- decoding each species of the current population to describe the architecture of the neural network.
- training each neural network with the architecture obtained in step one applying genetic algorithm for selecting the weights of the connections.
- evaluating the fitness of each species (of the coded architecture), based on the results of the training, i.e. based on the least total root-mean-square error of training.
- reproducing species in line with the chosen selection method.
- applying operators of crossing and mutation to obtain new generation.

2.6 Methods for Solving Multi-Objective Optimization Problem of Neural Network Models

Classical methods of multi-objective optimization (additive convolution method,[23] method of restrictions,[23] target programming,[24] minimax approach,[25] etc.) are hardly suitable for solving complicated problems. Coordinating several independent optimization cycles to obtain the Pareto optimal set can require excessive computations. Thereat, some methods can be sensitive to the character of the Pareto optimal front. Besides, some additional information may be required that would be hard or impossible to obtain.

Applying evolutionary approach makes it possible to overcome the disadvantages intrinsic to classical methods of multi-objective optimization. Apart from the above, evolutionary algorithms possess a number of advantages as follows: they ensure the global review of the solution space; they provide the possibility to avoid local minimums and to obtain several alternative solutions within one optimization cycle.

All mentioned above makes it possible to consider the practicability of applying evolutionary approach for solving multi-objective optimization problem of selecting the efficient ANN structure.

Applying evolutionary algorithm for multi-objective optimization stipulates that two major problems should be solved. First, it is necessary to choose such methods of assigning fitness values and selection that would help achieve the Pareto optimal set. Second, the issue should be solved of how the population is to be diversified in order to prevent the premature convergence and to achieve well-distributed and leveled undominated set.[26]

There are different concepts for assigning fitness values and selection in the process of solving the multi-objective optimization problem. Some approaches consider criteria separately, some approaches are based on the classical convolution method and some methods directly apply Pareto domination concept.

In the first case, in the process of selecting the individuals, the algorithm switches between the criteria according to some certain scheme. For instance, equal parts of temporal population can be filled based on different criteria.[27] Under another arrangement, the individuals can be compared based on some certain or based on random set of criteria.[28] According to Kursawe,[29] each criterion is assigned with a probability value that predetermines whether this criterion should be used at the next step of selection and whether it should be preset by the user or chosen randomly. However, all abovementioned approaches tend to so-called "extreme" solution and are sensitive to a non-convex Pareto optimal front.[30]

To overcome the abovementioned obstacle, in his studies Goldberg[31] expostulated on the idea of assigning a fitness value to an individual based on Pareto domination. The author has suggested an imperative procedure for ranking individuals: initially, all primarily undominated individuals are assigned rank "one", and they are temporarily removed from the population. Then, the next group of undominated individuals is assigned rank "two", and so on. Eventually, the fitness value of the individual is predetermined by the rank. The advantage of such approach is that the fitness values are assigned based on the quality of the whole population, while other methods use independent assignments of fitness values to each individual. Later, other researchers have suggested new schemes for assigning the fitness values, based on Pareto domination principle.[30,32] Though this class of evolutionary algorithms is capable of finding any Pareto optimal solution, its operation can be affected by the dimensions of the search space.

There are many approaches to solving the multi-objective optimization problem applying evolutionary methods, just to name a few: the methods based on mass selection with parameter variations, extension of definition, re-initialization of population, convolution, elitism and the whole range of such approaches based on the diversity of the population or "niching", as leveling the fitness values, restricted crossing, isolation.

To solve the multi-objective optimization problem of selecting the effective ANN structure, this study made use of the evolutionary algorithm FFGA (Fonseca and Fleming's genetic algorithm).

Method FFGA[32] represents a procedure of ranking individuals based on Pareto domination. Thereat, the rank of each individual is predetermined by the number of the individuals that dominate over this individual.

With FFGA method, the fitness of each of the species of population is assigned according to the algorithm as follows:

Input: P_t (population)
Output: F (fitness values)

Step 1: For each $i \in P_t$ the rank is calculated as follows:
$r(i)=1+|\{ j \mid j \in P_t \wedge j \; \mathbf{f} \; i \}|$.

Step 2: Population is sorted according to rank r. Each $i \in P_t$ is assigned with raw fitness $F'(i)$, interpolating from better individual ($r(i)=1$) to worse individual ($r(i) \leq N$); this algorithm applies linear ranking.

Step 3: Fitness values $F(i)$ are calculated by averaging the values of raw fitness $F'(i)$, among individuals $i \in P_t$ with identical ranks $r(i)$ (leveling fitness values in target space).

Here, symbol $|\cdot|$ signifies the power of the set. The individual, whose solution vector is undominated in relation to $m(P)$, has tank 1. Temporal population is filled stochastically, based on the rank value.

Later, this basic concept has been further developed; for example, today the leveling of fitness values is used together with the continuous introduction of random immigrants.

3. Results of Experimental Study

To verify the efficiency of the suggested approach for solving the problem of neural network modeling, a number of computational experiments have been carried out fulfilling the task of selecting the effective structure and adjusting the weights of ANN for practical purposes of modeling the melting process in the ore thermal furnace.[33]

Output parameters of the neural network model were represented by the characteristics of the process that describe its most important technological, energy-related and economical aspects and that make it possible to evaluate the efficiency of the ore thermal furnace operation:

- loss of non-ferrous metals to dump slag (CNI);
- productivity (P);
- electric power specific consumption (W_y);
- slag temperature (T);
- air emissions (G_{gas}), and other.

Principal management actions were represented by the parameters that affect the process considerably; thereat, they were the parameters which current values could be measured effectively given the complexity of obtaining the continuous, reliable and comprehensive information under the high temperature and aggressive environment of the furnace. Those are the parameters that characterize the charge of the furnace by separate components together with some electricity-related parameters.
Charged material parameters:

- amount of sinter (G_{sint});
- amount of silica (G_{sio2});
- amount of coke (G_C);
- amount of converter slag ($O_{c.s.}$);

Electrical parameters:

- input electric power (P);
- submersion of electrodes (H_e);
- voltage (U), etc.

Thereat, the chemical composition of the charged sinter and the chemical composition of the converter slag were assumed to be constant.

Dimensionality of the vector of the input effects on the process is equal to 10, that of the vector of the output parameters is also equal to 10. For the purposes of the experiments, a sample of 47 points was used. The parameters of the genetic algorithms applied to solving this problem are presented in Table 1 and Table 2.

Table 1. Parameters of genetic algorithm for training neural network

Maximum weight value	5
Minimum weight value	-5
Sampling interval	0.0001
Number of specimens in the population	100
Number of generations	100
Type of selection	Tournament
Number of individuals in tournament	10
Type of crossing	Equally probable
Type of mutation	Average

Table 2. Parameters of genetic algorithm for selecting structure of neural network

Number of inputs in neural network	10
Number of outputs in neural network	10
Maximum number of hidden neurons	10
Number of specimens in the population	40
Type of selection	Tournament
Number of individuals in tournament	5
Type of crossing	Equally probable
Type of mutation	Average
Number of activation functions	8

These parameters have been selected as the best of those found in the course of performing a series of the experiments focused on selecting the appropriate parameters.

This problem was solved as an optimization problem in two settings:

1. the problem of single-objective optimization, minimizing the root-mean-square error of ANN adjustment;
2. the problem of multi-objective optimization represented in (5).

Upon solving the problem as the single-objective optimization, the following averaged results have been obtained for 100 operations: average error of ANN adjustment – 0.113 (3.49%); average ANN computational complexity – 486.17.

For the multi-objective optimization setting, 100 experiments have been carried out too. Characteristic approximation of the Pareto set, for 80% of operations, includes 16 neural models with the adjustment error of 3.41% to 3.74% and with the computational complexity being in the range of 357 to 370.

Approximation of the Pareto front at the last generation of the genetic algorithm is shown in Figure 1.

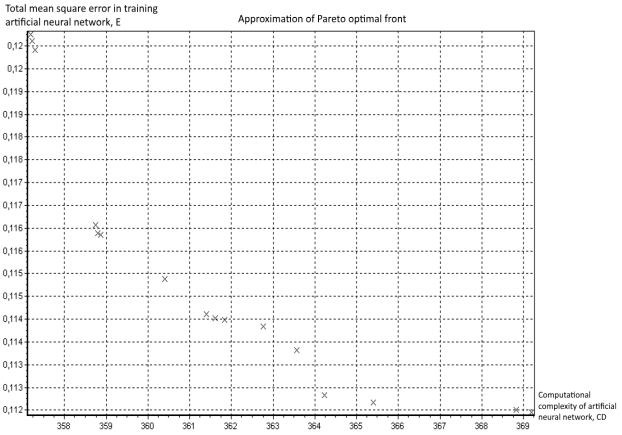

Figure 1. Approximation of Pareto optimal front for modeling melting process in ore thermal furnace.

4. Discussion

As a result of solving the task of neural network modeling for many times, the suggested approach proved it possible for a decision maker to choose from multiple options, and it also helped alleviating the complexity of neural network structures in the course of optimization.

The resulting approximation of the set of Pareto optimal solutions is quite representative encompassing wide spectrum of neural network structures: from the structures with low computational complexity but with relatively high approximation error up to the slow neural networks of good quality.

Comparing these results to those obtained with solving the problem of modeling in the single-objective setting, it becomes possible to distinguish a negligible difference in the approximation error together with the considerable decrease in the complexity of the structures of circa 25%.

This decrease in complexity makes it possible not only to improve the speed of the responses in the neural network model in its further practical application, but also to decrease the speed of its optimization, because all training algorithms require that the calculations should be performed for the neural network.

By contrast to the already known approaches to multi-objective optimization of the neural network models, this study suggests that the complexity of the model should be evaluated not in the course of its training, but in the course of establishing its structure. Thereat, the criterion should be represented not by the number of connections between the neurons that, no doubt, affect the speed of the network operation, but by the computational complexity that also depends on the time spent on calculating the activation functions at neurons.

5. Conclusion

Finally, it can be concluded that modern technologies for the neural network modeling and their practicability for solving complex problems of intellectual data analysis and management have been considered in detail within the framework of this study.

To select an appropriate method for solving the problems of structural and parametric synthesis of ANN, the comparative analysis of the existing approaches has been carried out, which resulted in justifying the application of the genetic algorithm that has been adapted for the purpose of solving the problems of unconditional optimization in adjusting the parameters of the neural network and in selecting the efficient structure of multi-connected ANN.

To obtain less computationally complex structures of multi-connected ANN without any loss to the training precision, the multi-objective optimization problem of the structure of the neural network model has been set and formalized, given its computational complexity.

The results of solving the practical problem have been presented; and they prove that applying the suggested approach makes it possible to alleviate the computational complexity of the obtained structures of ANN and also to help a decision maker select the neural network model among various options, given the preset precision requirements and given the available computational resources.

6. References

1. Rutkovskaya D, Pilinskiy M, Putkovskiy L. Neural networks, genetic algorithms and fuzzy systems: Translated from Polish. Moscow, Hot Line - Telecom, 2004.
2. Kruglov VV, Borisov VV. Artificial neural networks. Theory and Practice. Moscow, Hot Line - Telecom, 2002.
3. Gorban AN, Educating neural networks. Moscow, SP ParaGraf, 1990.
4. Gorban AN, Rossiev DA. Neural networks on personal computer. Novosibirsk, Nauka, 1996.
5. Komashinskiy VI, Smirnov DA, Neural networks and their application in control and communication systems. Moscow, Hot Line -Telecom, 2003.
6. Mirkes EM. Neural computer, draft standard. Novosibirsk, Nauka, 1999.
7. Aleksandr I, Morton H. An Introduction to Neural Computing. London, Chapman & Hall, 1990.
8. Girosi FT, Jones M, Poggio T. Regularization theory and neural network architecture. Neural Computation. 1995; 7:219–70.
9. Hassoun M. Fundamentals of Artificial Neural Networks. Cambridge, MA, MIT Press, 1995.
10. Muller B, Reinhardt J. Neural networks. Springer-Verlag, 1990.
11. Ossovskiy S. Neural networks for data processing: Translated from Polish by I.D. Rudnitskiy. Moscow, Finance and statistics, 2002.
12. Anderson D, McNeill G, Artificial neural networks technology. DACS report. 1992.
13. Rastrigin LA. Random search. Moscow, Znaniye, 1979.
14. Bartlett P, Downs T. Training a neural network with a genetic algorithm. Technical Report, Dept of Electrical Engineering. University of Queensland. 1990; 54–68.
15. Batishchev DI. Genetic algorithms for solving extreme problems. Textbook. Voronezh, VFTI, 1995.
16. Sergeyev SA, Makhotilo KV. Genetic algorithms in synthesis of straightforward neural networks. 13th International Conference, New information technologies in science, education and business. Proceedings. Ukraine, Crimea, Yalta - Gurzuf, 1996. p. 338–42.
17. Serov VA, Belov AV, Kholba YY. Synthesis of parameters of neural controller in adaptive control system based on genetic algorithms of multi-objective optimization. 4th International symposium, Intellectual systems. Proceedings. MIFI. Moscow. 2000; 87–9.
18. Adewuya A. New methods in genetic search with real-valued chromosomes. Master's thesis. Cambridge, Massachusetts Institute of Technology. 1996; 115–29.
19. Booker L. Improving search in genetic algorithms. Genetic algorithms and Simulated Annealing. London, Pitman. 1987; 61–73.
20. Haupt RL, Haupt SE. Practical Genetic Algorithms. 2 ed. Wiley, 2004.
21. Janikow CZ, Clair DS. Genetic algorithms simulating nature's methods of evolving the best design solution. IEEE Potentials. 1995 Oct; 39(14):31–5.
22. Hopfield J, Tank D. Neural computations of decisions in optimization problems. Biological Cybernetics. 1985; 52:141–52.
23. Cohon J. Multiobjective Programming and Planning. New York, John Wiley, 1978.
24. Steuer RE. Multiple Criteria Optimization. New York, John Wiley, 1986.
25. Koski J, Oscyczka A. Multi-criteria Design Optimization. Springer-Verlag, 1990.
26. Srinivas D. Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation. 1995; 2(3):39–44.
27. Schaffer JD. Multiple objective optimization with vector evaluated genetic algorithms. An International Conference on Genetic Algorithms and Their Applications. Proceedings. Pittsburgh, PA. 1985. p. 93–100.
28. Fourman MP. Compaction of symbolic layout using genetic algorithms. The First International Conference on Genetic Algorithms and Their Applications. Proceedings. Hillsdale, NJ, Lawrence Erlbaum. 1985. p. 141–53.
29. Kursawe F. Breeding ES - first results. Seminar on Evolutionary algorithms and their applications. 1996.

30. Horn J, Nafpliotis N, Goldberg D. A niched Pareto genetic algorithm for multiobjective optimization. The First IEEE Conference on Evolutionary Computation. Proceedings. Piscataway. 1994; 1:82–7.
31. Goldberg D. Genetic algorithms in search, optimization and machine learning. Reading, MA, Addison-Wesley, 1989; 230–41.
32. Fonseca CM, Fleming PJ. Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Parts I, II: A unified formulation. Technical report 564. Sheffield, UK, University of Sheffield, 1995 Jan.
33. Gonebnaya OE. Expert system of iron ore melting: PhD thesis. Krasnoyarsk, GUTsMiZ, 2004.