

# Evolving polynomial neural network by means of genetic algorithm: some application examples

Vasechkina, E.F. & Yarin, V.D.

*Marine Hydrophysical Institute of National Academy of Sciences,  
2 Kapitanskaya St., Sebastopol, 99011, Ukraine*

Email: [evasvyar@bios.iuf.net](mailto:evasvyar@bios.iuf.net)

## Abstract

The new technique for evolving feed-forward NN architecture by self-organising process is proposed. The approach doesn't require learning as separate process since evaluation of weights is carried out simultaneously with the architecture construction. The NN architecture is built by adding hidden layers to the network, while configuration of connections between neurons is defined by means of GA. GA runs for each neuron searching its optimal set of connections with the preceding layer. The output of the neuron is represented as polynomial function of the inputs with coefficients evaluated using the least-squares method. The proposed method was applied in a suite of tasks. It allows evolving compact feed-forward NN structures giving good decision to different real world problems and does not require large computational costs. It is able to generate an equation directly expressing the algebraic dependence of the output on the inputs. So it can be successfully applied to different simulation problems arising in biology, ecology and other natural sciences.

## 1. Introduction

Regression problems are usually handled using multilayered neural network (NN) architecture. Networks trained using the Back Propagation algorithm are a common choice for these types of problem. For standard Back Propagation the weights and biases are found using a gradient descent procedure but the architecture of the NN is found by experimentation. Definition of the NN architecture is of great importance, since it determines performance of the learning process and the network on the whole. Many works have appeared recently, where various techniques of network structure construction were proposed. Design of a network architecture using different evolutionary algorithms is interpreted as a search in the architecture space, where each point represents some network structure. In the works of Miller et al. (1989) and Stepniewski & Keane (1996) network connections were represented as a

binary matrix, where the entries in each row and column represent incoming and outgoing connections of each neuron respectively (“1” meant presence of a connection between nodes, “0” – its absence). This matrix is transformed into a vector by concatenating its rows, and the resulting vector is optimised using Genetic Algorithm (GA). Sase et al. (1993) proposed an encoding of a NN structure by a bitstring divided into two parts – the first part indicating presence or absence of neurons in the network and the second part resulting from the linearization of the connectivity matrix of the network.

Another approach was proposed by Harp et al. (1989) and Harp & Samad (1991). It is based on modular representation of a network structure, which is considered as a set of blocks, each block representing a group of neurons, perhaps, a layer or even several layers. The genotype sequentially includes these blocks separated by special markers and some additional “projection fields”, which describe interconnections between blocks. An approach suggested by Arena et al. (1993) is a two-dimensional representation of the NN of a fixed size, where a binary matrix encodes presence or absence of a neuron. In this case NN interconnections between active neurons of adjacent layers are assumed known beforehand. Sato & Ochiai (1995) interpreted the network as oriented graph and proposed special two-dimensional crossover operator performed by swapping the subgraphs connected to a common selected neuron.

Usage of each of these techniques assumes training of the network as a separate process, necessary for all variants of architectures to estimate their performance. It is computationally very expensive. It seems that a more efficient algorithm could evolve the architecture and the weights of the NN simultaneously. Such methods were proposed in a number of studies. Some of them (Dasgupta & McGregor 1992; Maniezzo 1994) were implemented as an extension of the above approaches by including NN weights in the genotype. With these techniques the search space in the evolutionary algorithm increases dramatically, making it much more difficult to find an optimal structure. It was found that design of special evolutionary operators depending on encoding is necessary to overcome emerging computational difficulties. Constructive algorithms, such as the Cascade Correlation algorithm introduced by Fahlman (1989) and further developed by Simon N. et al. (1992) and Phatak D.S. & Koren I. (1994), determine both the architecture and the parameters of the NN. They can reduce the training to single - node learning hence substantially simplifying and accelerating the training process. Compared to Back Propagation they have at least an order of magnitude improvement in learning speed. In Pujol (1999) a new method based on a two-dimensional representation is suggested. It includes operators for simultaneous evolving NN architecture and connection weights. This method is a general-purpose procedure able to design feed-forward and recurrent architectures but needs large computational resources.

In the present work we propose a simpler technique for evolving a feed-forward NN architecture with simultaneous identification of weights. It is based on a combination of ideas developed by Ivakhnenko (1971), Ivakhnenko et al. (1994) (so called Group Method of Data Handling - GMDH) and GA. One of the functions of GMDH, which combines a set of techniques, is to build multiple-regression equations for dependent variables belonging to a complex system with uncertain interconnections and often provided by insufficient and noisy data. This multilayer algorithm is essentially a sort of a polynomial NN. GA has been involved in it to search an optimal set of connections between neurons in adjacent layers. The following section is devoted to a brief review of GMDH methods. The proposed approach is considered in the third section. And finally, some applications to real-world problems are discussed.

## 2. Group Method of Data Handling

The method is based on the sorting-out procedure, which implements consequent testing of models chosen from a set of models-candidates in accordance with the given criterion. Using a supervised learning procedure this method allows finding functional dependence of the output on the most significant inputs of the system. Most GMDH algorithms use the polynomial basic functions. Other non-linear functions, such as finite-difference, logistic, harmonic, can be also used. General connection between input and output variables is expressed by the Volterra functional series, a discrete analogue of which is Kolmogorov-Gabor polynomial:

$$y = a_0 + \sum_{i=1}^M a_i x_i + \sum_{i=1}^M \sum_{j=1}^M a_{ij} x_i x_j + \sum_{i=1}^M \sum_{j=1}^M \sum_{k=1}^M a_{ijk} x_i x_j x_k,$$

where  $X(x_1, x_2, \dots, x_M)$  - vector of input variables;  $A(a_1, a_2, \dots, a_M)$  - vector of weights.

Components of the input vector  $X$  can be independent variables, functional forms or finite difference terms. As the available data sample is often small and insufficient to estimate all regression coefficients at once, GMDH suggests an iterative procedure to overcome this difficulty. For this purpose a set of model-candidates (particular descriptions) are generated in accordance with the specific iterative rule. These models compete between themselves for a chance to be selected for the next step of the procedure. Selection of models-candidates is based on the external criterion; this is an essential feature of GMDH. Available data sample is divided into three parts: training, checking and testing. Training subsample is used for calculation of weights by means of the least-squares method (LSM). Checking one is used for estimation and selection of models-candidates and testing - for a final choice of the model.

Solution of practical problems led to development of a number of GMDH algorithms. They mainly differ by a choice of the models-candidates generator, accepted basic functions and external criteria used. We'll consider two main modifications of GMDH, which are broadly used for regression type problems.

### 2.1 Combinatorial GMDH Algorithm

The main Combinatorial GMDH algorithm has a multilayered iterative structure. Its specific feature is that the iteration rule (particular description) does not remain constant but expands with each new layer. In the first layer all the models of the simplest structure in the form

$$y = a_0 + a_1 x_i, \quad i = 1, 2, \dots, M$$

are subject to sorting and a number  $F$  of the models that are the best by the criterion are selected.

In the second layer models of a more complex structure are sorted. These models are constructed for output variables of the best models of the first layer:

$$y = a_0 + a_1 x_i + a_2 x_j, \quad i = 1, 2, \dots, F; j = 1, 2, \dots, M; F \leq M$$

In the third layer the structure of the model-candidate is:

$$y = a_0 + a_1x_i + a_2x_j + a_3x_k \quad i = 1,2,\dots,F ; j = 1,2,\dots,F ; k = 1,2,\dots,M$$

and so on. Build-up of the layers continues as long as the criterion minimum decreases. For large values of the freedom of choice  $F \rightarrow M$ , the algorithm ensures a complete sorting of all the models.

## 2.2 Iterative Multilayered GMDH Algorithm

In the Iterative Multilayered algorithm, the iteration rule remains the same for all layers. For example, a particular description in the form

$$y = a_0 + a_1x_i + a_2x_j + a_3x_ix_j$$

is used in the first layer, a particular description

$$z = b_0 + b_1y_i + b_2y_j + b_3y_iy_j$$

in the second layer, a particular description

$$w = c_0 + c_1z_i + c_2z_j + c_3z_iz_j$$

is used in the third layer, and so on. In other words, the output values of a preceding layer serve as arguments in the succeeding layer. The algorithm checks  $2^d-1$  possible models for each two variables  $x_i$  and  $x_j$ , where  $d$  is an amount of terms in a particular description (in our example  $d = 4$  as the set of basic functions is  $\{1, x_i, x_j, x_ix_j\}$ ). For all possible pairs of input variables we must evaluate  $(2^d-1)M(M-1)/2$  models, each one with its own value of the criterion used. The number of calculations grows tremendously as the vector of input variables enlarges. However, with this iteration rule some models may be missed, which may result in the so-called multilayeredness error. To avoid missing useful models it is necessary to make additional efforts.

Comprehensive testing of GMDH implemented by Dolenko et al. (1996) proves that it is a powerful tool for mathematical modelling that can be used to solve a wide variety of different real-life problems. The most pronounced feature of GMDH is that it can choose the really significant input variables among dozens of these, thus actually reducing the dimension of the solved problem. However, they notice that GMDH is hardly suitable for very large problems with a great number of inputs which are nearly equally significant. One more useful property of the method is its ability to generate a formula directly expressing the algebraic dependence of the output on the inputs. This allows one to find out some important relations among the input variables: for example, whether there are cross-correlations between some variables, etc.

It is easy to see that selection of candidate models in GMDH is analogous to search of incoming connections of the neuron in a feed-forward NN. We propose to organise this search by means of GA, as it has proved its high performance when applied to difficult optimisation problems with unknown fitness landscapes.

### 3. Combined Algorithm Description

The NN architecture is built by adding hidden layers to the network, while configuration of each neuron connections is defined by means of GA. An elitist GA with binary encoding, fitness proportional selection, standard operators of crossover and mutation is used in the algorithm.

Available data for network learning is represented as a table with columns including a series of matching input and output parameters.<sup>1</sup> A row of this table we'll call a "data table point" or simply a "data point". Let us denote the system output  $Z$  and the corresponding input vector  $U$ . The activation function of a neuron is defined as an identity function; i.e. a neuron output  $Y_n$  is represented as a sum

$$Y_{nk} = a_{n0} + \sum_i^F a_{ni} X_{ik} + \sum_i^F \sum_j^F a_{nij} X_{ik} X_{jk} \quad (1)$$

where  $F$  is an amount of nodes in the preceding layer;  $a_{n0}$  is the bias of the neuron;  $a_{nij}$  are weights of inputs with indices  $i, j$ ;  $X_{ik}$  are values generated by neurons of the preceding layer connected with the "n" node;  $k$  is the row index in the data table.

Such formulation is rather universal, since it corresponds to a Taylor expansion restricted to quadratic terms. Let's consider a neuron output  $Y_n$  as a particular solution of our problem. Indeed, if the current layer were the last layer of the network, numerical values generated by this node would be some possible network output. These values are required to be as close as possible to the target values of the output  $Z_k$ . To this effect it is necessary to compose and solve a linear system of normal equations constructed by substituting  $X_{ik}$  and  $Y_{nk}$  from the data table in Eq. 1 and averaging them.<sup>2</sup> The solution of normal equations is a set of weights corresponding to connections of the "n" node. Therefore, in a sample of training data used each neuron output (or particular solution)  $Y_n$  is "attracted" as closer as possible (by criterion of the least-squares error) to exact solution of the problem. Because of insufficiency of data one can't estimate all the coefficients in Eq. 1 simultaneously, that's why it is necessary to organise some sorting-out procedure to calculate them step by step. Besides, it is possible to expect that not all inputs  $X_i$  will be equally important and some of them should be excluded to fit the target function best.

---

<sup>1</sup> Data table columns could include, for example, time series of state vector components of the dynamic system under consideration.

<sup>2</sup> Note, that in the first layer  $X_{ik}$  are interpreted as numerical values of the network inputs  $U_{ik}$ . In subsequent layers  $X_{ik}$  denote the values generated by neurons of the preceding layer. Target function is always the system output  $Z$ . Thus the data table is regularly overwritten using outputs of neurons in a preceding layer as inputs for the succeeding layer when the network evolves.

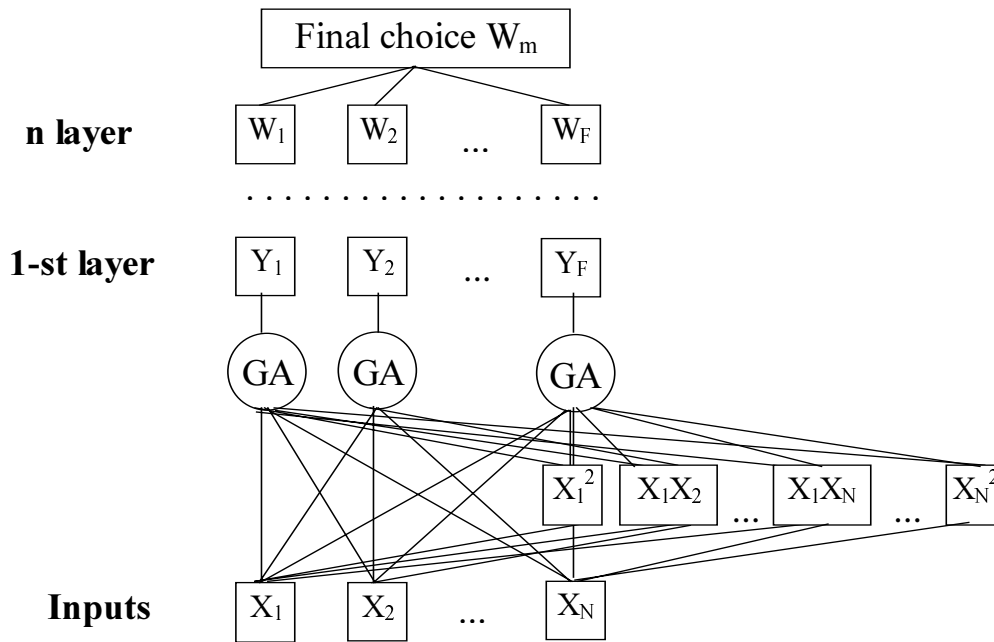


Figure 1. Network representation. Each of the layers is equivalent to the first.

Unlike GMDH, which uses deterministic search, we propose to define configuration of connections between adjacent layers by means of GA. The genotype of an individual of the population is defined as a bitstring  $L = (001100010...01)$ , where “1” means presence of the correspondent term in the eq.(1), and “0” – its absence. In the linear variant of the algorithm the position of ‘1’ in the genotype denotes presence of the connection between this node and the corresponding one in the preceding layer ( $a_{nj} \neq 0$ ). So each individual of the population represents some configuration of incoming connections of the neuron. In general, presence of  $a_{nij} \neq 0$  in front of a nonlinear term in Eq. 1 according to ‘1’ in the genotype generates several connections between nodes in the adjacent layers at once. Figure 1 schematically represents the network structure evolving in such a way. GA runs for each neuron searching its optimal set of connections with the preceding layer. During learning neurons are able to estimate which inputs are necessary to minimize fitness of the neuron. Thus, the neurons are able to organise the network structure by themselves and may be called “active neurons”.

To design a network able to generalize we introduce a rule to estimate fitness of an individual on a sample that hasn't been used for identification of weights. For this purpose all available sample is divided into three parts: training, checking and testing. Weight coefficients corresponding to “1” in the genotype are evaluated in the training set by LSM. For this purpose a system of normal equations is generated and solved. Fitness is estimated in the checking sample as relative mean-square error (RMSE) (mean-square error (MSE) divided by standard deviation of the output  $Z$ ). Having repeated this procedure for each of the neurons, which amount is restricted to some maximum number, we get a configuration of potential connections of the layer with the previous one. The same procedure is repeated for other layers, which amount is also restricted to some maximum value.

To regulate the self-organising process of building the network it is necessary to define an accuracy criterion. We introduce this criterion as the best fitness of the neuron calculated in the layer. This value determines the number of layers in the network. The build-up of the NN is stopped when the accuracy criterion reaches the minimum. At the last layer of the network there is still a set of nodes. Each of these nodes represents some variant of the task solution with the corresponding network architecture. Final choice of the network structure is carried

out on the testing sample, which up to this moment was not utilized in the search procedure. The sequence  $Y_{mk}$  giving minimum RMSE of the target function in the testing sample defines a choice of the output node “ $m$ ”, and at the same time a final choice of the network architecture.

For implementation of the self-organising process of evolving NN with active neurons it is necessary to define the following:

- the number of inputs and relevant data table for network learning;
- maximum size of the network, i.e. maximum allowable amount of layers and neurons in a layer;
- population size and maximum allowable number of generations.

Often it is necessary to perform a complementary tuning of the network structure. Experiments have shown that network performance depends on location in the table of data points used for NN weights calculation and of those used for fitness evaluation. We propose to use GA for selection of data points for training and checking sequences. For this purpose all data sample excluding testing subsample is sequentially numbered, and the genotype of an individual of the population is defined as a set of point numbers. At the beginning of the procedure the population is generated at random. The phenotype is constructed using the best neuron structure obtained earlier for the first network layer. Fitness function can be evaluated as RMSE calculated in the checking sample or otherwise according to specificity of the task. The overfitting problem, which can arise upon implementation of this procedure in the nonlinear algorithm, can be overcome by additional calculation of fitness in the testing sample. This value shouldn't be larger than the fitness calculated in the checking sample.

## 4. Application Examples

Previously the authors used analogous approaches for solving a number of tasks emerging in research of marine environment. For example, in Yarin et al. (1999) the problem of marine ecosystem model identification was considered and solved using GMDH. In this paper we present three examples that illustrate applications of linear and nonlinear variants of the proposed algorithm to different real-life problems.

### 4.1 Pole-balancing system

This is a well-known benchmark for evolving neural controllers considered in a number of studies, for example, Whitley et al. (1995) and Pujol (1999). The task is to balance a pole hinged in the center of a moving cart by applying a force to the cart. The pole can move in a vertical plane and the cart moves in a one-dimensional track. The controller must provide a sequence of left and right pushes to the cart, in order to keep the system within specified limits. The only forces applied to the system are a control force  $f$  and gravity. The mass is uniformly distributed along the pole and its thickness is negligible compared with the length.

Under these assumptions equations describing this system are:

$$\ddot{x} = \frac{f + mL(\dot{\theta}^2 \sin \theta - \ddot{\theta} \cos \theta)}{m_s} \quad \ddot{\theta} = \frac{m_s g \sin \theta - \cos \theta (f + mL\dot{\theta}^2 \sin \theta)}{\frac{7}{3} m_s L - mL \cos^2 \theta} \quad (2)$$

where  $x$  and  $\theta$  are coordinates of the cart and the pole counted from the null position, points from above mark their velocities and accelerations,  $m$  is the pole mass = 0,1kg,  $m_s$  is the mass of all system = 1,1kg,  $L$  is half the length of the pole = 0,5m,  $g$  is the gravity acceleration. The cart under the force calculated by the controller can move within the boundaries of  $\pm 2,4$ m. The deviation angle of the pole shouldn't exceed  $\pm 360$ .

It is necessary to build a neuron structure able to generate a sequence of values of the force  $f$  in Eq. 2 allowing balancing a pole during a fixed number of time steps. Coordinates and velocities of the pole and the cart are inputs to this NN; the only output is the control force. In this work we consider the case of a continuous control force only. The proposed algorithm needs a data sample for its implementation. The necessary sequence of control force values was generated using GA. During 200 time steps the system was controlled by the force generated by GA and resulting time series were used as a data sample for NN training. GA ran at each time step and searched an optimal value of the force necessary to hinder the pole from falling down. The genotype of an individual of the population represented a real number  $f$  in binary notation. Fitness of an individual was estimated as:

$$Fit_i = \left| \frac{x_i}{x_{max}} + \frac{\theta_i}{\theta_{max}} \right| + 0.1 \left[ \left( \frac{\dot{x}_i}{\dot{x}_{max}} \right)^2 + \left( \frac{\dot{\theta}_i}{\dot{\theta}_{max}} \right)^2 \right] \quad (3)$$

where  $x_i$  и  $\theta_i$  are positions of the cart and the pole after application of the force calculated by GA. Normalization factors are marked with the index *max*, these are maximum available coordinates of the pole and the cart. Although the velocities were not restricted by task specification, for their normalization we used 5 m/s and 270  $^{\circ}$ /s accordingly. Initial values used for generation of the training data sample were  $\theta_0 = 32^{\circ}$ ,  $d\theta_0/dt = 36^{\circ}$ /s,  $x_0 = -2,4$  m,  $dx_0/dt = -1,5$  m/s.

Resulting time series of four input parameters and the force were gathered up into the data table for network training. Sample division was the following: the testing sample consists of 40 last data points, training and checking – of 80 points each. The linear variant of the algorithm was applied to the task of NN construction; in most runs it resulted in a one-layer network with four inputs and one output without hidden layers. RMSE in the checking subsample was 8-10% and 10-13% in the testing one. However, designed network structures did not provide the complete task solution, namely, control of the pole-balancing system at all sets of initial states represented in Table 1.

Table 1. Initial states for the pole-balancing system.

Pole position ( $^{\circ}$ )	0	29,8	27	-20	0	27	0
Pole velocity ( $^{\circ}$ /s)	-65	65	0	-65	0	0	0
Cart position (m)	2,4	-2,4	0	0	2,14	-2,4	-1,8
Cart velocity (m/s)	0	0	0	0	0	0	-0,75

To improve the accuracy of the NN weights there was implemented the above-mentioned procedure of “complementary tuning”. All points of the data sample were numbered and the genotype was defined as the sequence of the points' numbers. For each individual of the population the weight coefficients within one-layer network structure (4 – 1) were estimated in the training subsample and the NN resulted was used for control of the system. Fitness of an individual was evaluated as the number of time steps in which the pole and the cart could be kept in limited boundaries. Most “difficult” for control initial conditions were chosen to run the procedure.

Usage of all available data for calculation of weights by LSM didn't bring satisfactory results. That's to say, information included in the full sample was superfluous for building an optimal NN structure. Usually a required set of data points was searched rather quickly. Already within the second generation of the population there were one or more individuals allowing performing the task. So the one-layer network with refined weights was able to control the system during 1000 time steps under all given initial conditions. It is necessary to note, that after implementation of "complementary tuning" RMSE of the resulted network became larger than before and rose up to 18-25% in the checking and testing subsamples. This affirms the known statement that an optimal model is not the most precise model in a limited sample. Compared to results of analogous experiments cited in Pujol (1999) the NN structures obtained with the proposed algorithm were simpler, since they had no hidden layers.

The linear variant of the algorithm applied to this task give comparable results with linear regression, if available data set is enough to estimate all coefficients of the equation. It proves out its advantage when applied to the task poorly provided with data, especially, if an input vector includes a great number of variables, which are not equally significant. The practical application of the algorithm has shown that for most identification problems, arising in research of natural dynamic systems, linear representation is insufficient and application of the nonlinear variant of the algorithm is required.

## 4.2 Flutter

The task was to find a neuron structure that could generate an output that is as closer as possible to the target function  $Y$  using input values  $X$ . Benchmark data presented in De Moor B.L.R., DaISy [96-008] were used for testing the proposed technique (Fig. 2). The set of input variables was constructed from  $X$  using time delays (0,-1,-2,...-10,-15,-20,-25,-30).

For network evolving there were used 500 data points, per 225 points in the training and checking samples and 50 points in the testing one. The whole length of the data table was 770; remaining data points were used for estimation of performance of the method.

All of the NN architectures evolved had rather complex configuration, the number of layers varied between 1 and 4, maximum amount of neurons in each layer was 15. Figure 3 represents results of the target function fitting in the part of data sample that wasn't used for NN construction. This model has RMSE in the testing sample 16% and in the whole sequence that wasn't used for network training - 33%. For comparison, results of the target function fitting using linear regression are also presented in Fig. 3. RMSE of this approximation found upon the same data set is 92%.



Figure 2. Input and output signals of the "Flutter" benchmark.

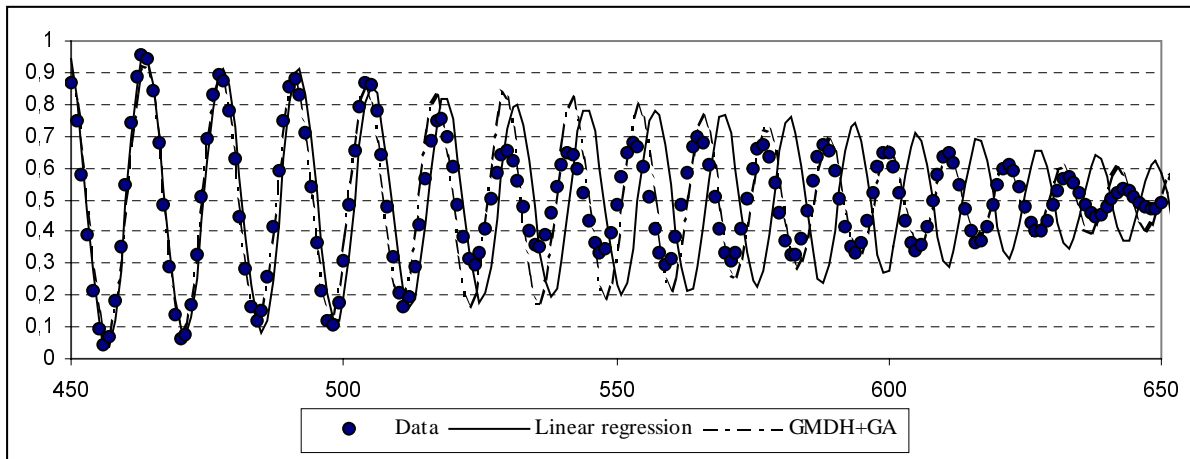


Figure 3. Comparison of linear regression with the approximation found by means of combined algorithm.

### 4.3 Estimation of functional dependencies between active layer parameters in the Black Sea.

The goal was to search empirical dependencies between surface and averaged values of chemical and biological parameters in the active layer of the Black Sea. Actual data collected by research vessels of the Marine Hydrophysical Institute and the Institute of Biology of the Southern Seas over a period of years were used in the calculations. Hydrological stations including a complete set of observations were selected from the Data Bank of the Black Sea (1996). Also marine meteorological observations were used. The Black Sea was schematically divided into three regions: the North-West shelf bounded by the latitude of  $44^{\circ}36'$  N, and the West and the East deep-sea regions bordered upon one another along the longitude of  $35^{\circ}$  E. Selected stations were grouped according to their location and the season when the observations were carried out. Thus prepared data were used for generation of a statistical ensemble for each parameter observed. Using the appropriate statistical ensemble there were calculated the first and the second probability moments of the vertical sections of each parameter. Empirical orthogonal functions (EOF) found upon the covariance matrix were used for reconstruction of missing data and necessary smoothing of profiles<sup>3</sup>. Corrected profiles were then averaged in the layer of 0-100m. Averaged parameters together with their surface values were placed into the data table for NN training.

According to the problem statement all variables were classified as inputs or outputs. We implied by inputs those variables that can be measured with comparative ease (meteorological and sea surface parameters) or calculated using hydrodynamic model. The outputs were chemical and biological components (dissolved oxygen, phosphate and chlorophyll "a") averaged in the layer of 0-100m. The data table for NN training was formed in such a way that each row incorporated input and output data on one hydrological station.

Applying the above technique we found target dependencies between the variables of the system. By way of illustration we adduce here an equation for the average oxygen content with absolute MSE in the testing sample 0.23ml/l and RMSE - 22%. This formula was

<sup>3</sup> Reconstruction of missing data was implemented by means of technique described in Vasechkina & Yarin (2000).

obtained by extracting an analytical expression from a simple two-layer NN structure built by applying the nonlinear variant of the algorithm:

$$O_s = 27.6550 - 0.0275T_{av}^2 - 0.0588S_{av}^2 - 6.88 \cdot 10^{-4}T_s S_{av} - 0.65 \cdot 10^{-4}V^3 - 1.42 \cdot 10^{-4}V^2(T_a - T_s)^2 \quad (5)$$

Here  $O_s$  - average oxygen content in the layer of 0-100m,  $T_s$ ,  $S_s$ ,  $T_{av}$ ,  $S_{av}$  - surface and average temperature and salinity accordingly,  $V$  - wind speed,  $T_a$  - air temperature.

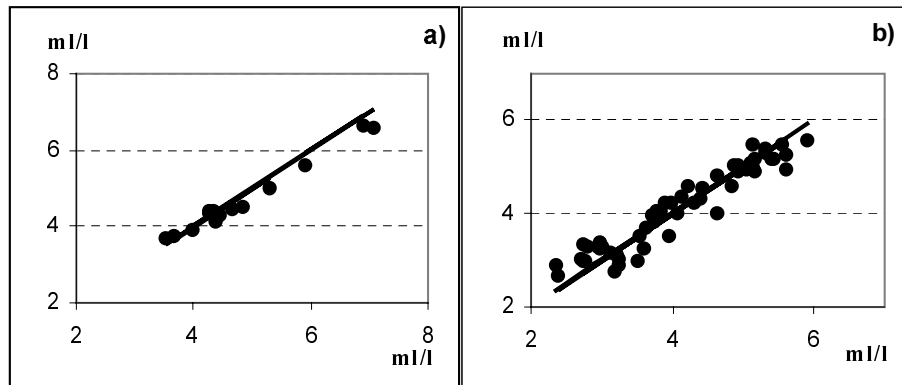


Figure 4. Comparison of the model predicted oxygen concentration with actual data: a) testing subsample of spring data; b) independent sample of autumn data.

This functional dependence was found by evolving the NN upon spring data from the West deep-sea region of the Black Sea. The data table included 97 points, 17 of them were used as a testing sample. To estimate generalizing power of this formula it was applied to autumn data in the same region. After slight tuning of coefficients (free term basically) the model gave satisfactory results presented in Fig. 4. The left side of this figure (Fig. 4a) represents target function fitting in the testing subsample of spring data. On the right side (Fig. 4b) there are results of approximation obtained on the autumn data. Actual data are plotted on the abscissa and modelled values - on the ordinate. The solid line indicates a set of points with zero approximation error. The autumn sample included 50 data points; RMSE of its approximation by using Eq. 5 was 30%. This formula has no “consuming part”, in other words, there are no terms dependent upon oxygen-consuming components of the system. These factors can't be considered as input parameters since they are poor provided with observations and, on the other side, are limited themselves by oxygen. For phosphate and chlorophyll “a” there were also built the NNs and found the appropriate regression equations:

$$Ch_{av} = 1.1118 - 5.4910 \cdot 10^{-2}T_{av} - 5.0396 \cdot 10^{-1}O_s - 1.4732 \cdot 10^{-3}T_{av}O_{av} + 4.1154 \cdot 10^{-3}S_s^2 - 1.0511 \cdot 10^{-2}S_sO_{av} + 3.0216 \cdot 10^{-2}O_s^2 + 7.1855 \cdot 10^{-2}O_sCh_s - 2.9836 \cdot 10^{-2}O_{av}Ch_s + 2.3953 \cdot 10^{-2}Ch_s^2, \quad (6)$$

$$PO_{4(av)} = 202.6137 - 28.0632\sigma_{av} + 0.9841\sigma_{av}^2 + 1.914 \cdot 10^{-3}T_sS_s - 9.048 \cdot 10^{-3}S_{av}\sigma_{av}, \quad (7)$$

where  $Ch$ ,  $O$ ,  $PO_4$ ,  $\sigma$  denote chlorophyll “a”, oxygen, phosphate and water density;  $av$  and  $s$  indices designate sea surface values and those averaged in the layer of 0-100m.

Approximations of target functions in the testing sample by means of these equations are represented in Fig. 5. RMSE in the testing sample was 31% for phosphate and 14.5% for chlorophyll "a" (MSE – 0.15ml/l and 0.05ml/l correspondingly). The NNs were trained on the spring data sample; the nonlinear variant of the algorithm was used for evolving network structures.

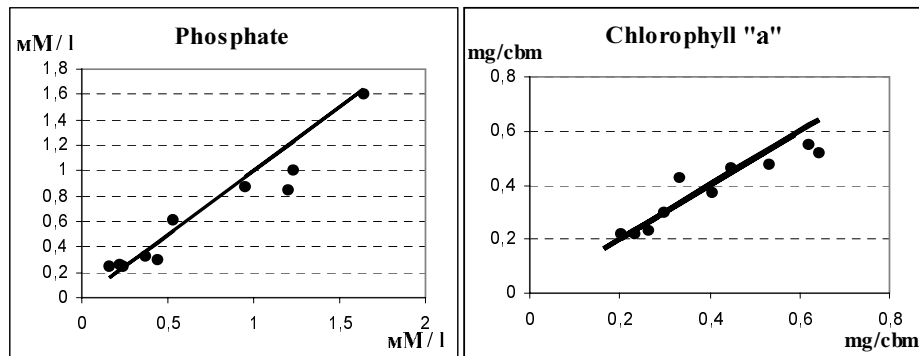


Figure 5. Comparison of model predicted phosphate and chlorophyll "a" concentrations with actual data.

## 5. Conclusion

The new method of NN architecture design with simultaneous evaluation of weights was proposed and tested in a suite of real-world problems. This approach allows evolving compact feed-forward NN structures and does not require significant computational resources. It is of great importance that the structure of the network is defined by a self-organising process based on the least-squares method and constructed network does not require learning as a separate process. The suggested approach can provide linear and nonlinear approximations of target function according to the task features. Mean-square errors of such approximations usually do not exceed 15-30% from standard deviation of the target function. In our estimation this approach can be successfully applied to different problems arising in biology, ecology and other natural sciences.

## References

- Ash T. (1989), Dynamic node creation in backpropagation networks, *Connection Science 1*, 365-375.
- Arena P., Caponetto R., Fortuna I., & Xibilia M. (1993), (M.L.P.) Optimal topology via Genetic Algorithms, *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, 670-674.
- Dasgupta D. & McGregor D. (1992), Designing application-specific neural networks using the structured Genetic Algorithm, *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*, IEEE Computer Society Press, 87-96.
- Data Bank of the Black Sea (1996), MHI of National Academy of Sciences, Ukraine, URL: <http://www.mhi.iuf.net/DEPTS>.

- De Moor B.L.R., DaISy [96-008]: Database for the identification of systems, Department of Electrical Engineering, ESAT/SISTA, K.U.Leuven, Belgium, URL: <http://www.esat.kuleuven.ac.be/sista/daisy/>, 8.02.2000. Used dataset: Wing flutter data, Mechanical Systems.
- Dolenko S.A., Orlov Yu.V., & Persiantsev I.G. (1996), Practical implementation and use of Group Method of Data Handling (GMDH): Prospects and Problems, *Proceedings of ACEDC'96*, PEDC, University of Plymouth, UK
- Fahlman S. & Lebiere C. (1990), The cascade correlation architecture, *Advances in Neural Information Processing Systems*, 2, 524-532. Morgan Kaufman, San Mateo, CA.
- Ivakhnenko A.G., Ivakhnenko G.A., & Muller J.A. (1994), Selforganisation of neuronets with active neurons, *Pattern Recognition and Image Analysis*, 4, 177-188.
- Ivakhnenko A.G. (1971), Heuristic self-organising systems in cybernetics, Technique, Kiev.
- Maniezzo V. (1994), Genetic evolution of the topology and weight distribution of neural networks, *IEEE Transactions on Neural Networks*, 5(1), 39-53.
- Miller G., Todd P., & Hegde S. (1989), Designing neural networks using Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, 379-384.
- Phatak D.S. & Koren I. (1994), Connectivity and performance tradeoffs in the cascade correlation learning architecture, *IEEE Transactions on Neural Networks*, 5, 930-935.
- Pujol J. (1999), Evolution of artificial neural networks using a two-dimensional representation, Ph.D. thesis, School of Computer Science, University of Birmingham.
- Stepniewski S. & Keane A. (1996), Topology design of feed-forward neural networks, *Proceedings of the Parallel Problem Solving from Nature*, 771-780.
- Sase M., Matsui K. & Kosugi Y. (1993), Inter-generational architecture adaptation of neural networks, *Proceedings of the International Joint Conference on Neural Networks*, Nagoya, Japan, 3, 2941-2944.
- Sato Y. & Ochiai T. (1995), 2-D Genetic Algorithms for determining neural network structure and weights, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, San Diego, CA, USA.
- Simon N., Cororaa H., & Kerckhoffs E. (1992), Variations on the cascade correlation learning architecture for fast convergence in robot control, *Proceedings of the Fifth International Conference on Neural Networks and their Applications*, Nimes, France, 455-464.
- Whitley D., Gruau F., & Pyeatt L. (1995), Cellular encoding applied to neurocontrol, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan-Kaufmann, 460-467.
- Vasechkina E.F., Yarin V.D. (2000), Using Genetic Algorithms in mathematical simulation of ecosystems, *Proceedings of the International Workshop "Environmental Control Systems – 2000"*, Sebastopol, Ukraine.
- Yarin V.D., Vasechkina E.F., Timchenko I.E., & Igumnova E.M. (1999), Model of the Azov Sea ecosystem dynamics, *Marine Hydrophysical Journal*, 4, 53-63.