HUNGARIAN JOURNAL
OF INDUSTRIAL CHEMISTRY
VESZPRÉM
Vol. 30. pp. 241 - 245 (2002)

CLASSIFICATION NEURAL NETWORKS IMPROVEMENT USING GENETIC ALGORITHMS

A. WOINAROSCHY, V. PLESU and K. WOINAROSCHY

(Department of Chemical Engineering, University Politehnica of Bucharest,
1-5, Polizu Street, Bucharest 78126, ROMANIA)

Received: April 8, 2002

The aim of the present work consists the development of a procedure capable to realize a high accurate classification neural network with a reduced number of neurons. In order to avoid local minima, the weights of a proposed three-layer network were computed using a common genetic algorithm. The performances of the genetic algorithm were strongly improved by several means that make an increased balance between exploration and exploitation of the search space. Thus, in order to decrease the number of genes, respectively weights, the dimension of the output vector was minimized by identification of classes with binary numbers. A very favorable effect was obtained by seeding the initial population with a good chromosome obtained by the use of the classical "delta-rule" learning procedure. It was also investigated the effect of the initial population size, the bounds imposed to the weights of the inter-neuronal connections, the number of neurons in the hidden layer, fitness expression, etc.

Keywords: neural networks, genetic algorithms, classification

Introduction

An important field of neural network applications in bioprocessing and chemical engineering consists in classification problems, like process fault detection and diagnosis, detection and location of gross errors in experimental data sets, spectral analysis, models discrimination and identification of model parameters, etc. There are several types of neural networks used for classification problems: back-propagation, radial-basis-function, learning-vector-quantization, probabilistic networks, networks based on adaptive-resonance-theory, and so on. Each type has some advantages and consequently disadvantages, depending on the nature of the problem. Representatives for chemical and biochemical processes are the problems with non-uniform decision regions where the data points of each class are scattered. For these problems radial-basis-function and back-propagation networks perform better [1,2]. The use of gradient of the error function during the learning stage affects the performances of these networks due to ending into a local minimum. In fact, the gradient technique is an example of a hill-climbing strategy, which exploits the best solution for possible improvement; on the other hand, it neglects exploration of the search space. Random search is a typical example of a strategy, which explores the search space ignoring the exploitations of the promising regions of the space. For small spaces, classical exhaustive methods usually suffice; for larger spaces special artificial intelligence techniques must be employed. Genetic algorithms are among such techniques, being a class of general purpose (domain independent) search methods which strike a remarkable balance between exploration and exploitation of the search space [3]. For several years, genetic algorithms have been used to evolve the neural networks structure, as well as the weights of the inter-neuronal connections (e.g. see [4-7]). The aim of the present work consists in an extended investigation of the features of genetic algorithms in order to realize a high accurate classification neural network with a reduced number of neurons.

Methodology

The investigations were done on the three layers networks with linear transfer function for the input layer, and sigmoid transfer function for the hidden and output layers. Because the number of neurons in the input layer is imposed by the number of elements of the vectors that are classified, the structural variables of these networks

Table 1 The effects of several factors on the performances of chemical reactor fault-diagnosis network

S_{ip}	B	N_h	N_o	RMS error	Percentage of wrong classifications (%)
100	[-100; 100]	3	4	0.3478	27.20
200	[-100; 100]	3	4	0.3418	26.25
500	[-100; 100]	3	4	0.2974	25.00
500	[-50; 50]	3	4	0.3827	26.25
500	[-100; 100]	3	4	0.2974	25.00
500	[-250; 250]	3	4	0.4840	37.50
500	[-100; 100]	3	4	0.2974	25.00
500	[-100; 100]	5	4	0.2701	21.25
500	[-100; 100]	7	4	0.1845	16.25
500	[-100; 100]	9	4	0.2395	18.75
500	[-100; 100]	3	4	0.2974	25.00
500	[-100; 100]	3	3	0.1411	20.00
500	[-100; 100]	3	2	0.2398	21.25

equal the number of neurons in the hidden and output layers. The number of neurons in the output layer depends on the rule of classes codification. There are no theoretical guidelines to establish the number of hidden neurons. Different from the mentioned works, here each chromosome corresponds with a complete set of the weights of the inter-neuronal connections for a given structure of the network (respectively each gene represent a weight). The structural variables, respectively, the number of neurons in the hidden and output layers, were step-by-step modified in an external loop: the procedure starts with the minimum numbers of the corresponding neurons and these are progressively increased up to imposed limits.

The genetic algorithm used is a MATLAB implementation [8] that can be downloaded at ftp://ftp.eos.ncsu.edu/pub/simul/GAOT. Float representation of chromosomes has been used. The selection of candidate chromosomes for crossover and mutation is made according with a ranking selection function based on the normalized geometric distribution. Three types of crossover are applied: simple, interpolated, and extrapolated crossover. In the simple crossover, the crossover point is randomly selected. The interpolated crossover performs an interpolation along the line formed by the two parents. The extrapolated crossover performs an extrapolation along the line formed by the two parents in the direction of better parent. Four types of mutation are applied: boundary, multi-nonuniform, nonuniform, and uniform mutation. Boundary mutation changes one gene of the selected chromosome randomly either to its upper or lower bound. Multi-nonuniform mutation changes all genes, whereas nonuniform mutation changes one of the genes in a chromosome on the base of a non-uniform probability distribution. This Gaussian distribution starts wide, and narrows to a point distribution as the current generation approaches to the maximum generation. Uniform mutation changes one of the genes based on a uniform probability distribution. The numbers of applications of the different crossover and mutation operators are imposed as parameters of the genetic algorithm. The investigation of the effects of these parameters was out of the aims of the present work, and their default values have been used, respectively for each generation: 2 simple, 2 interpolated, and 2 extrapolated crossover, 4 boundary, 6 multi-nonuniform, 4 nonuniform, and 4 uniform mutation.

Due to the use of a maximization algorithm, the chromosome fitness corresponds to the negative value of RMS error. The attempts to use other expressions for chromosome fitness (e.g. the average relative error) did not give meaningful improved results.

Applications

Chemical reactor fault-diagnosis

This application and the corresponding data are presented in [9]. The input vector contains reactor inlet temperature, reactor inlet pressure, and feed flowrate. The elements of the output vector are three fault classes: low conversion, low catalyst selectivity, and catalyst sintering.
The following factors were studied:

a) the size of initial population (S_{ip});
b) the bounds imposed to the weights of the inter-neuronal connections (B);
c) the number of neurons in the hidden layer (N_h);
d) the number of neurons in the output layer (N_o);

The last factor corresponds to different rules of classes codification. There are four types of outputs, corresponding to four classes: the correct operating regime, and the three fault regimes. The classical codification corresponds to the use of four output neurons, as follows: [1 0 0 0]; [0 1 0 0]; [0 0 1 0]; [0 0 01]. Also, there are possible other two codification methods: a codification using three output neurons, respectively: [0 0 0]; [1 0 0]; [0 1 0]; [0 0 1], and a codification with two output neurons: [0 0]; [0 1]; [1 0]; [1 1]. The last codification is in fact the binary representation of each class number.

Our investigations in the field of classification neural networks indicated that the activity and corresponding response of the output neurones must close to 0 or 1. A neural network with one output neuron (having the smallest number of the inter-neuronal connections weights) for which the above mentioned 4 classes are codified with the activity domains: 0-0.25; 0.25-0.5; 0.5-0.75; 0.75-1 cannot be trained to give a correct classification.

Due to the random generation of the initial population of the genetic algorithm, each test was repeated five times. The results presented in the *Table 1* represents the corresponding average values. In all tests, for a correct comparison of the results, the total number of generations (the stopping criterion) was the same, respectively 1000.

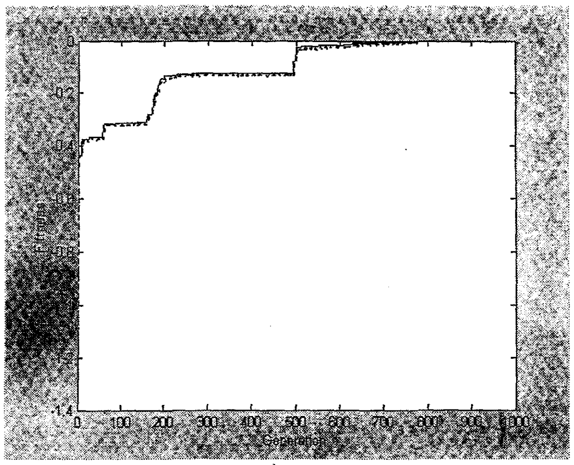

Fig.1 The evolution of mean (....) and best (—) fitness for the optimum neural network

The analysis of these results indicates that:

- Expansion of the initial population size has a favorable effect due to increasing the probability to generate individuals with a good fitness. Unfortunately, this effect is present only in simple cases, for well-defined problems with smooth response surface. For highly complex problems, the search space in which genetic algorithm usually operates is so large, that taking few additional chromosomes into initial population cause a very small, or no detectable effect, unless the problem surface is very smooth.
- A too small, or a too large range of the weights values has a non-favorable effect: in the first situation the optimization solution is arbitrarily restricted, and in the second case the search space is unreasonably enlarged.
- As in the case of gradient-based learning algorithms, there is an optimum number of neurons in the hidden layer for each neural network application, and the corresponding value must be establish through an iterative procedure.
- Similarly, the best rule of classes codification and the corresponding number of neurons in the output layer must be found by iterations.

With the best values of the investigated factors from *Table 1*, respectively $S_{ip} = 500$; $B = [-100; 100]$; $N_h = 7$; $N_o = 3$, and by the aid of the genetic algorithm the weights of the corresponding neural network were established. This exhibit excellent results: RMS error $= 0.007$ and no wrong classification (related to the training set, due to the absence in the original mentioned reference [9] of a test set). The corresponding evolution of the genetic algorithm is represented in *Fig.1*.

It is remarkable that the best values of RMS error obtained for this example by BAUGHMAN and LIU [9] (using the classical "delta rule" as learning procedure) at the end of 50,000 training iterations were: 0.088 for a network with 3 neurons in the hidden layer and sigmoid transfer function, and 0.037 for a network with 5 neurons in the hidden layer and hyperbolic tangent transfer function.

Consistency analysis of fuzzy sets

This case study was already presented in literature [10] as an application of a modified back-propagation neural network. The reason of the actual selection consists in the fact that the application represents a difficult classification problem involving non-linearly separable patterns.

In many chemical processes due to several reasons there are some variables that cannot be measured directly on-line (e.g. biological or catalyst activity). In these cases the values of measured variables are fuzzy sets. Related to system constraints and feasible domains of the variables these fuzzy sets must be consistent.

In order to use a graphic representation, a hypothetical example with two measured variables, x_1 and x_2, and one unmeasured variable, x_3, all restricted to a system of three strongly non-linear constrains, was considered:

$$0.69x_3e^{1.24x_1} - x_2 = 0 \qquad (1)$$

$$-0.216x_3 - 0.869x_1x_3 + 0.955x_1^2x_3 - x_2 = 0 \qquad (2)$$

$$-0.5x_3 + 0.275x_1x_3 + 0.261x_1^2x_3 + 0.18x_1^3x_3 + x_2 = 0 \qquad (3)$$

The feasible domains of the variables are:

$$-1 \le x_1 \le 1; \qquad -1 \le x_2 \le 1; \qquad -1 < x_3 < 1$$

The number of variables is incidentally the same as the number of constraints. Usually, the number of variables is greater than the number of constraints. The measured values of the variables correlated with the feasible domains of all variables, and with constraints' violations will generate several classes of errors, depending on combination of violated restrictions. For this application the following four classes were considered:

A - all restrictions are satisfied;
B - constraint (1) is violated regardless of constraint (2) and constraint (3);
C - constraint (2) is violated and constraint (1) and constraint (3) are respected;
D - constraint (3) is violated regardless of constraint (1) and constraint (2).

These four domains, the data from the training set (points numbered from 1 to 124), and test data (prefix T) are indicated in *Fig.2*. It can be observed that all learning data correspond to points placed on, or near the bounds.

The best results obtained by the aid of the genetic algorithm at the end of 500 generations are exposed in *Table 2*.

The best result obtained in [10] using the classical "delta rule" as learning procedure corresponds to a

Table 2 The best results obtained by the aid of the genetic algorithm with no-seeding and with seeding the initial population with a chromosome having RMS error = 0.1012 (S_{ip} = 500; B = [-130; 130]; N_h = 5; N_o = 2)

Procedure	Trial number	RMS error	Percentage of wrong classifications (%)
No-seeding	1	0.2816	25.00
No-seeding	2	0.2506	22.58
No-seeding	3	0.2466	20.77
No-seeding	4	0.1445	12.10
No-seeding	5	0.2446	20.97
No-seeding	Mean	0.2336	20.32
Seeding	1	0.0907	3.22
Seeding	2	0.0895	6.45
Seeding	3	0.0902	5.65
Seeding	4	0.0894	6.45
Seeding	5	0.0902	4.84
Seeding	Mean	0.0900	5.32

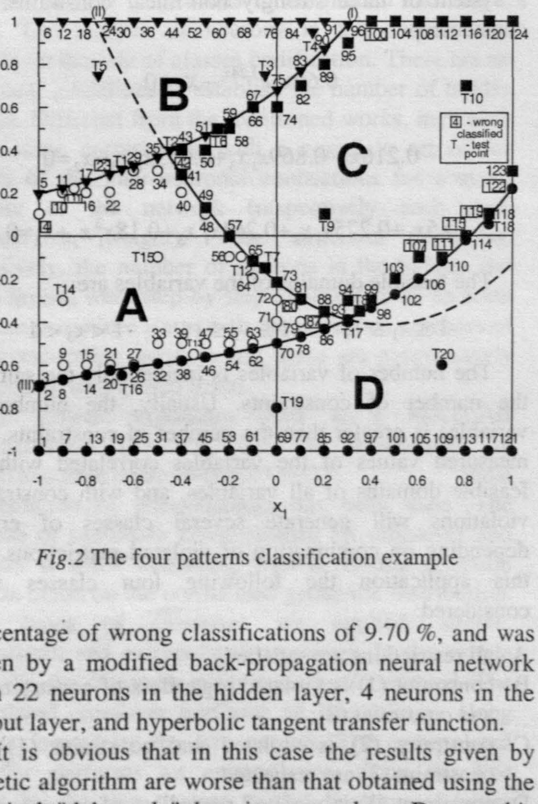

Fig.2 The four patterns classification example

percentage of wrong classifications of 9.70 %, and was given by a modified back-propagation neural network with 22 neurons in the hidden layer, 4 neurons in the output layer, and hyperbolic tangent transfer function.

It is obvious that in this case the results given by genetic algorithm are worse than that obtained using the classical "delta rule" learning procedure. Due to this fact, a hybrid procedure was proposed: in a first stage the search is made with the aid of "delta rule" learning procedure; next, in the second stage, the initial population of chromosomes is seeding with the best solution from the end of the first stage. The results obtained with this "seeding" procedure are encouraging. Thus, for the considered network, "delta rule" learning procedure gives the weights corresponding with a RMS error of 0.1012. This solution was improved with 10 % by the genetic algorithm. The results obtained with seeding the initial population are given also in *Table 2*.

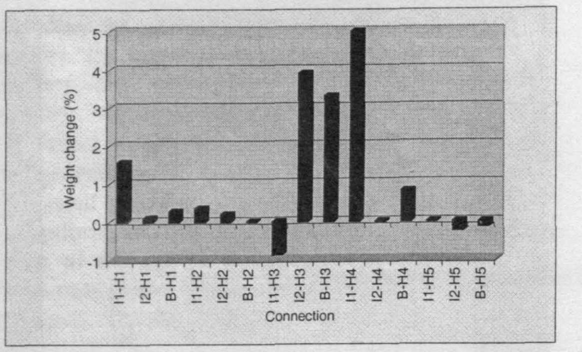

Fig.3 Changes of the weights of the inter-neuronal connections between the input and hidden layers (I – input; H – hidden; B – bias)

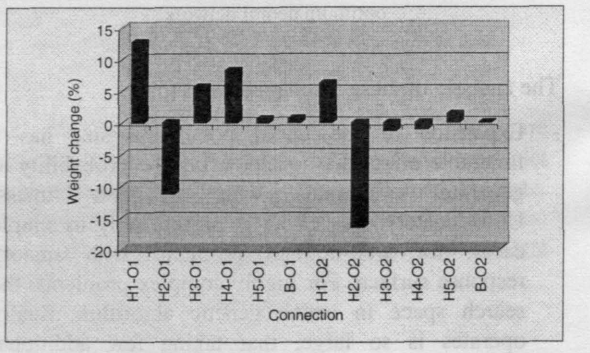

Fig.4 Changes of the weights of the inter-neuronal connections between the hidden and output layers (H – hidden; O – output; B – bias)

The improved solution was obtained mainly due to changes of the weights of the inter-neuronal connections between the hidden and output layers (*Figs.3* and *4*).

For this difficult classification problem, the mean percentage of wrong classifications given by this hybrid procedure is with 82 % better than the best solution corresponding to a larger neural network obtained in [10].

The corresponding evolutions of the genetic algorithm for "no-seeding" and "seeding" procedure are represented in *Fig.5*, and respectively *Fig.6*.

Conclusions

The genetic algorithms are an exciting way to realize high accurate classification multilayer neural networks. Different from other works, we search for the network structure in an external iterative loop. We consider that this two-stage procedure gives a better control of the results. Also, this procedure allows the selection of a new structural variable, respectively the number of neurons in the output layer. This means different rules of classes codification, and the equivalent modifications in the training data sets. To realize these in a single optimization loop seems to be a little bit complicated. Maybe the two-stage procedure is many computer time

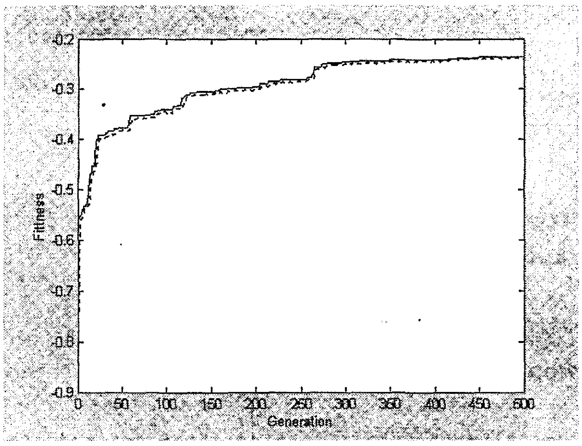

Fig.5 The evolution of mean (....) and best (—) fitness for "no-seeding" procedure

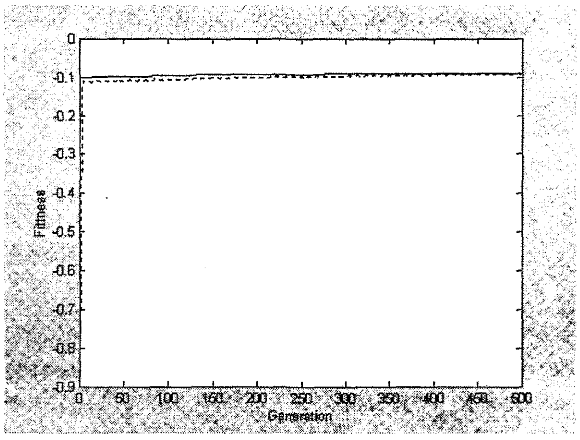

Fig.6 The evolution of mean (....) and best (—) fitness for "seeding" procedure

consumers, but our interest was focused on the accuracy of the results. It is obvious that, in order to realize high accurate classification multilayer neural networks for practical applications (e.g. process control or expert systems), the network performances are much more important than the computer time spent for this.

Another direction of investigation in this work was the performance of a hybrid algorithm composed from "delta rule" and genetic algorithms. In fact this hybridization was realized by seeding the initial population from the genetic algorithm with the best solution obtained with classical "delta rule" learning procedure. Despite the warning of some authors [11] that by seeding the genetic algorithm chases after a local minimum, and takes time to find its way out, very good results were obtained with this procedure in a difficult classification problem.

Our research, realized in the frame of two applications, finished with good results, so we can conclude that the use of the genetic algorithms represents a promising way to realize high accurate classification multilayer neural networks.

SYMBOLS

B	bounds imposed to the weights of the inter-neuronal connections
N_h	number of neurons in the hidden layer
N_o	the number of neurons in the output layer
S_{ip}	size of initial population
x_i	variables in Eqs. (1)-(3)

REFERENCES

1. ALDRICH C. and VAN DEVENTER J. S. J.: Chem. Eng. Sci., *1994*, 49(9), 1357-1368
2. ALDRICH C. and VAN DEVENTER J. S. J.: Ind. Eng. Chem. Res., *1995*, 34(1), 216-224
3. MICHALEWICZ Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer Verlag, Berlin, 1996
4. HARP S. A., SAMAD T. and GUHA A.: Towards the Genetic Synthesis of Neural Networks in: Davies, L. (Ed.): Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991
5. MANIEZZO V.: IEE Trans. Neural Net., *1994*, *5* (1), 39-53
6. GAO F., LI M., WANG F., WANG B. and YUE P. L.: Ind. Eng. Chem. Res., *1999*, 38(11), 4330-4336
7. BOOZARJOMEHRY R. B. and SVRCEK W. Y.: Comput. Chem. Engng., *2001*, 25(10), 1075-1088
8. HOUCK C. R., JOINES J. A. and KAY M. G.: NCSU-IE, Technical Report, North Carolina State University, 95-09, 1995
9. BAUGHMAN D. R. and LIU Y. A.: Neural Networks in Bioprocessing and Chemical Engineering, Academic Press, San Diego, 1995
10. WOINAROSCHY A., ISOPESCU R., NITA I. and DINU S.: Data Filtering via Artificial Neural Nets, 5th World Congress of Chemical Engineering, San Diego, vol. I, 1011-1016, 1996
11. HAUPT R. L. and HAUPT S. E.: Practical Genetic Algorithms, Wiley, New York, 1998