ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 185-192

A structural genetic algorithm to optimize High order neural network architecture

I. Chalkiadakis G. Rovithakis M. Zervakis

Dept. of Electronic & Computer Engineering
Technical University of Crete
73100, Chania, Greece

Abstract: A structural genetic algorithm is proposed to optimize the High Order Neural Network (HONN) architecture and the parameters of activation function. This work partitions the genes of chromosomes into control genes and parameter genes in a hierarchical form. Control genes binary encoded in the chromosome represent the activity of neurons, while real-valued parameter genes, represent the parameters of the sigmoid activation function. To verify the performance of our system, the HONN is used to approximate 2-D and 1-D functions.

1. Introduction

The use of neural networks (NN) in a wide area of applications has been well established. Predicting the optimal topology for a NN is a difficult task since choosing the neural architecture requires some a-priori knowledge and probably a lot of trial and error runs. Moreover, the topology of the neural network directly affects two of the most important factors of neural network training, generalization and training time [1]. Until now methods to obtain the optimal topology of a neural network generalized as:
a) The trial and error method where different topologies of neural networks are tested and then the smallest network that best performs selected. b) Destructive and constructive methods where initially assume a large or small network respectively and then prune off or add nodes or/and hidden layers [2, 3]. c) Using of natural selection such genetic algorithms or genetic programming, which choose the best network from a population of networks, considering as criterion a fitness value each chromosome scored.
This paper proposes a structural genetic algorithm technique [4, 5] to optimize High Order Neural Network (HONN) architecture [7] and the parameters of the activation function. The genes of chromosome are classified into control genes and parameter genes in a hierarchical form. Control genes binary encoded in the chromosome, represent the activity of neurons, while parameter genes, real valued, represent the parameters of the sigmoid activation function. The result of this method is that the computational effort and the optimization of the HONN are improved.
The paper is organized as follows: in section 2 the theory of high order neural network and of a learning algorithm are briefly reviewed. Section 3 introduces the structural

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 185-192

genetic algorithm optimizing the HONN. In section 4 we verify our system by approximating one and two-dimensional functions.

2. Problem statement

Let's consider a function y = f(χ), $y \in \Re$ representing the actual function, and let $\hat{y} = \hat{f}(x)$ be an approximation model of the actual function f(χ). Consider the approximation error as:

$$e = f(x) - \hat{f}(x) \tag{1}$$

Although we can directly measure the approximation error e, f(χ) is completely unknown for us. To overcome this uncertainty we consider the first order filter:

$$\dot{z} = -az + e \tag{2}$$

where $z \in \Re$ is the filter output and α is a design constant. In order to provide a mathematical acceptable solution for our problem we use Higher Order Neural Networks (HONN). HONN's are single layer fully connected nets, containing high order connections of sigmoid functions in their neurons of the form:

$$s(x) = \frac{m}{1 + e^{-l(x-c)}} - \lambda \ , \tag{3}$$

where the parameters m, l, represents the bound and the slope of sigmoid' s curvature and λ is a bias constant. Mathematically, the neural network can be expressed in the form:

$$\hat{y} = W^T S(x) \tag{4}$$

where $\hat{y}$ is the approximation model, W is a L-dimensional vector of adjustable weights, and finally S(x) is a L-dimensional vector which elements are in the form:

$$S_i(x) = s^i(x) \ , \qquad \text{i = 0, 1, 2, …L} \tag{5}$$

In the general case where **x** is a N dimensional vector, then the elements of the vector S(x) are in the form:

$$s^i(x_1) \cdot s^j(x_2) \ldots s^k(x_n) \quad \text{i ,j, …,k=0, 1, 2,…, L} \tag{6}$$

where s(x) is a monotically increasing smooth function, in most cases represented by the sigmoid function of the form (4). In other words, a HONN may be visualized as a full scale polynomial of sigmoid functions. In the above product we are not consider the case where i, j, …, k are simultaneously 0. HONN model is known for it's excellent approximation properties further discussed in [6, 7]. The Higher order neural network architecture that we use in our application presented in Figure 1. In this topology, in order to optimally approximate the candidate function, we choose the maximum order L of HONN to be 6, and the maximum number of the regions where **c** can take their possible values, to be 5. This pictured as a topology that has the form of five parallel neural structures incorporated in one neural network. In order to establish stable learning laws Lyapunov theory applied, a process described extensively in [7,8]. The system of differential equations used in order to train the HONN is:

$$\dot{z} = -az + (y - \hat{y}) \tag{7}$$

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 185-192

and

$$\dot{W} = -\gamma \cdot W + z \cdot s(x_i) \tag{8}$$

In the above equations α and γ are design constants and W are the connection weights updated through the training process.

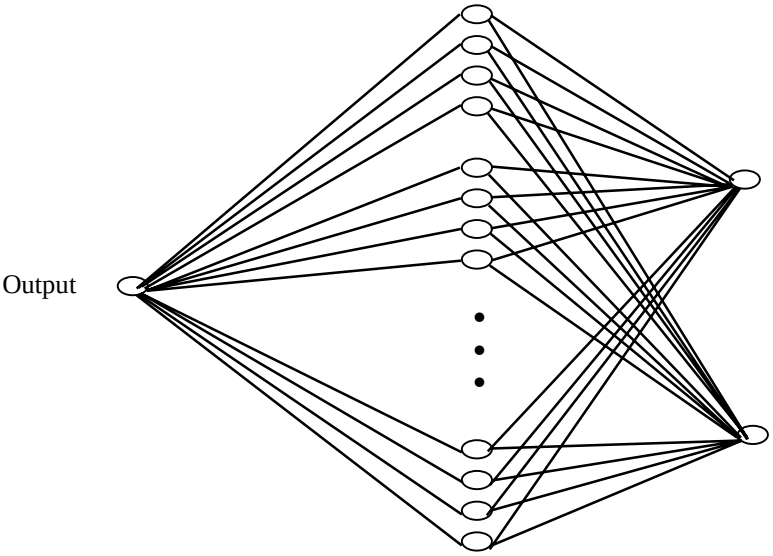

Figure 1 Higher Order Neural Network architecture

3. A structural genetic algorithm to optimize HONN's architecture

Our paper proposes a structural genetic algorithm technique [4, 5] in finding an optimal architecture for the HON network. Simultaneously the genetic algorithm searches for the optimal parameters related with the activation function of the HONN and the parameters α, γ related with the learning process. This approach partitions the genes of the chromosomes in connection genes and parameter genes. Connection genes are binary encoded in the chromosome structure and indicate the activity or not of a connection in the network. Parameter genes, real-encoded in the chromosome structure, represent a real parameter of the sigmoid activation function and two parameters related with the training algorithm. The advantage of this approach is that the connection genes are classified in a hierarchical form, which is the ideal formulation for the genes as the neurons, (or a network of neurons), can be formed within the string of chromosome [4].

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 185-192

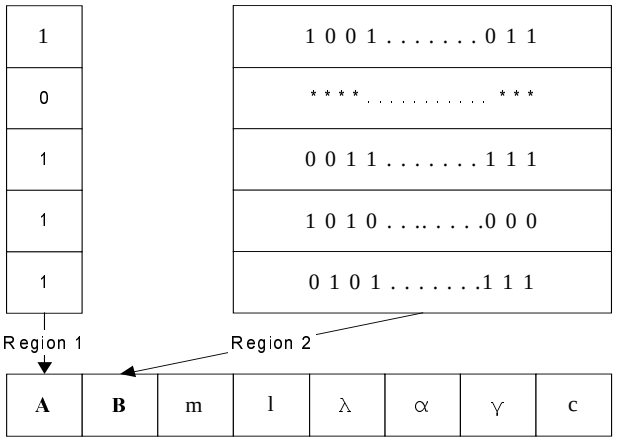

Figure 2 Chromosome structure

This technique decreases the computational effort, decreasing simultaneously the searching time in the binary space, resulting in searching much more network topologies in a predefined period.
Figure2 presents the chromosome architecture. Each one partitioned in three domains. The first and the second incorporate the control genes, while the third region incorporates the parameter genes. Control genes, divided in two main categories: the first consists of the binary values of domain 1, which indicate the activity or not for each one of the five parallel neural structures. The second category consists of binary values of domain 2, indicating the activity or not of each neuron in each structure. Parameter genes, represent the parameters of sigmoid activation function and two design parameters (α and γ) related with the training algorithm. A "*" indicates a 'don't care symbol'. This means that if a '0' is presented in the first layer, no computation process is performed at the related sub-network. The complete form of the genetic system presented in the block diagram of Figure 3. This system has the ability to search for an optimal HONN topology, defining simultaneously optimal parameters for the activation function of the network. In this system, for each topology examined with genetic search, a parallel process is used to train the HONN, defining the connection weights. Each cycle of the genetic search is known as a generation. In order to initialize the vectors **A** and **B** of the chromosome a binary generator used. This generator initialize the elements of the vectors, giving them the values "0" or "1" with probability 50%. Then the analog part of the chromosome must be initialized with random real values, considering the restrictions that m, l, a, γ>0. The next step in the process is to compute the fitness value of each chromosome. Each chromosome represents an integrated network structure, and a parallel training process take place for each one. After the end of the learning process, testing samples are introduced into each one of the trained networks-chromosomes and the objective function is evaluated:

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 185-192

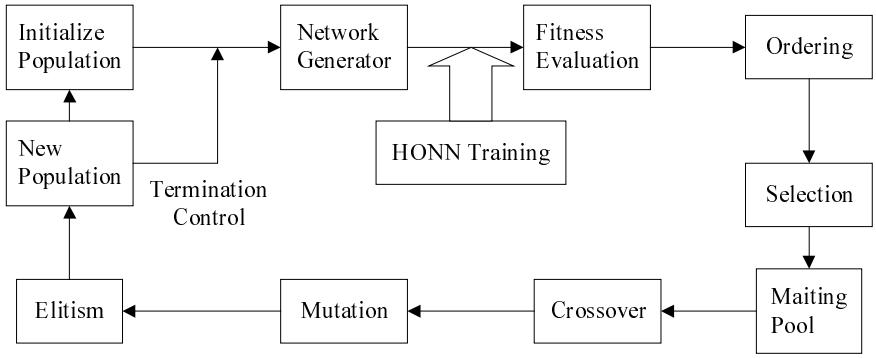

Figure 3 Block diagram of the genetic system

$$F = \frac{ct}{1+f} \tag{9}$$

where ct is a constant value and f is the accuracy function defined as:

$$f = \sum_{i=0}^{N} |y - \hat{y}| \tag{10}$$

where N is the number of testing samples.

After the chromosomes are assigned with fitness values, they are sorted, and chromosomes scored in the top 40% are designated as the candidate parents of the next generation. All the other networks are disregarded [9].

Since there are two types of genes encoded in the chromosome, binary and real, specific genetic operators designed to suit for their purposes. In this work a modified crossover operator used. The couples of parents are randomly selected from the mating pool. Each chromosome is divided into three regions (see Figure 2). Crossover is to be applied in each of the above regions separately with probabilities p_1, p_2, p_3 respectively. Two real numbers r_1, r_2 are randomly selected. If r_1 is larger than p_1, a one-point crossover is to be applied in the genes of region 1. Bit strings are separated in two parts by a randomly defined crossover point. Exchanging the parts of the two parents creates the new chromosomes. If r_2 is larger than p_2, one point crossover is to be applied in the genes of region 2, as shown in Figure 4. A kind of internal crossover operator is applied in this case. Each gene consists of 5 binary strings and we randomly select two couples. The next step is to randomly select a crossover point and the new gene created by exchanging the parts of the binary strings.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 185-192

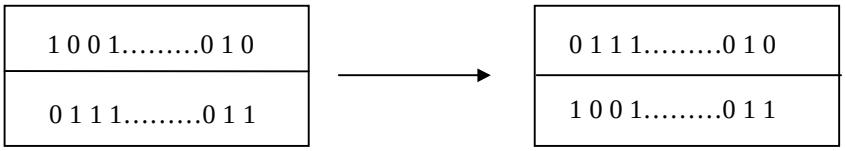

Figure 4 Crossover operation on control genes of region 2

The next step is to apply crossover operator in the analog part of the chromosome. A binomial crossover operator used in this case [10]. For each real valued parameter of a children chromosome, the value is taken from the first parent with probability p_3, and from the second parent with probability $(1 - p_3)$. It is important to mention here that it is possible the three crossover operators to act simultaneously on the candidate chromosome.

A modified mutation operator may act with probability p_m. For each chromosome in the mating poola real number r is randomly selected. If r is less than p_m, mutation operation is applied. Each one of the three parts of the chromosome has the same probability of mutation to operate on it. In the binary part of the chromosome one bit from the string selected randomly and change it's value. Mutation on the analog part of the chromosome operates by randomly selecting a parameter thus adding random noise. The difference in the value of the parameter is given by the equation:

$$\Delta x = 0.5 \cdot (r) \cdot x \tag{11}$$

where r is a random number in the interval [-0.5 0.5].

Finally the chromosome with the highest fitness value is copied to the new population created (Elitism criterion). In such a way a local search is performed around the best chromosome of each generation [11].

4. Simulation results

To verify the performance of the proposed system two functions are used: The first function has the form $f(x)=x^3+x+0.5$ were $-3<x<2$. With the proposed method the approximate error defined by: $E = \frac{1}{N}\sum_{1}^{N}(y - \hat{y})^2$ (12) is reduced to $4.5 \cdot 10^{-3}$. The second $f(x_1, x_2)=s(x_1)+s(x_1)s(x_2)+s^3(x_1)s(x_2)+s(x_1)s^2(x_2)+s^3(x_2)$ where s(x) given by equation :

$$s(x) = \frac{2}{1+e^{-x}} - 1 \tag{13}$$

In this system a number of 100 chromosomes are searching to find the optimum solution with total number of generations 1500. In each generation the 30% of the top scored chromosomes are considered as candidate parents for the genetic operations. Crossover probability for each couple is 40%. Mutation probability for each chromosome in the mating pool is 30%.

In this application we choose the maximum order of the network to be 6, and c take its values from 5 different spaces. Finally, input vector's dimensionallity is 2. These result in a network structure which contain 240 hidden units. The genetic algorithm has to optimize 267 different parameters in order to define an optimum architecture for the HONN. The two dimensional function is finally approximated with an error defined by equation (12) that does not exeeds $1.677 \cdot 10^{-6}$. Figures 5, 6 present how the performance of the genetic algorithm evolves with the number of generations.

5. Conclusions

A hierarchical genetic algorithm is proposed to optimize High order neural network topology and to define optimal parameter values for the activation function. Genetic system have to tune a large number of parameters in order to approximate optimally a candidate function and considering the results presented above, our system is effective.

References

[1] A. Blanco, M. Delgado, M. C. Pegalajar "A genetic algorithm to obtain the optimal recurrent neural network" Int. Journal of approximate reasoning vol.23 2000.
[2] M. Mozer, P. Smolensky "Skeletonization: A technique for trimming the fat from a network via relevance assessment" Advances in Neural information Proc. Systems I (pp 107-115) 1989.
[3] D. Chen, C. Giles, G. Sun, H. Chen, Y. Less, M. Goudreau "Constructive learning of recurrent neural network" IEEE Int. Conf. on Neural networks vol.3 1993.
[4] K. S. Tang C. Y. Chan, K. F. Man, S. Kwong "Genetic structure for NN topology and weights optimization" Genetic algorithms in engineering systems: Innovations and applications, conf. publication no.144 IEE 1995.
[5] K. F. Man, K. S. Tang, S, Kwong "Genetic algorithms" Springer publications 1998.
[6] E. B. Kosmatopoulos, M. Polycarpou, M. A. Christodoulou, P. A. Ioannou "High Order neural network structures for identification of dynamical systems" IEEE Trans. on Neural networks vol.6 no.2 pp.422-431, 1995.
[7] G. Rovithakis, M. Christodoulou "Adaptive Control with recurrent High order neural network" Springer 2000.
[8] G. Rovithakis, M. Christodoulou "Adaptive control of unknown plants using dynamical neural networks" IEEE Trans. on Systems man and cybernetics vol. 24 no.3, pp. 400-412, 1994.
[9] P. Angeline, G. Saunders, J. Pollack "An evolutionary algorithm that constructs recurrent neural networks" IEEE Trans. on NN. Vol.5 no1 1994.
[10] Kalyanmoy, Deb "Genetic algorithms for function optimization" Genetic algorithms and soft computing – Physica Verlag.
[11] V. Petridis, E. Paterakis, a. Kehagias "A hybrid neural – genetic multimodel parameter estimation algorithm" IEEE Trans. on neural networks, vol.9 no.5 1998.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 185-192

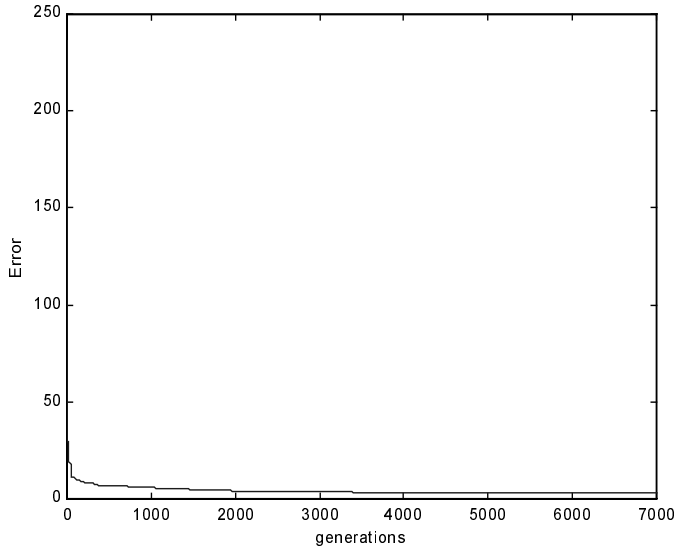

Figure 5 Performance of the genetic algorithm for $f(x)=x^3+x+0.5$

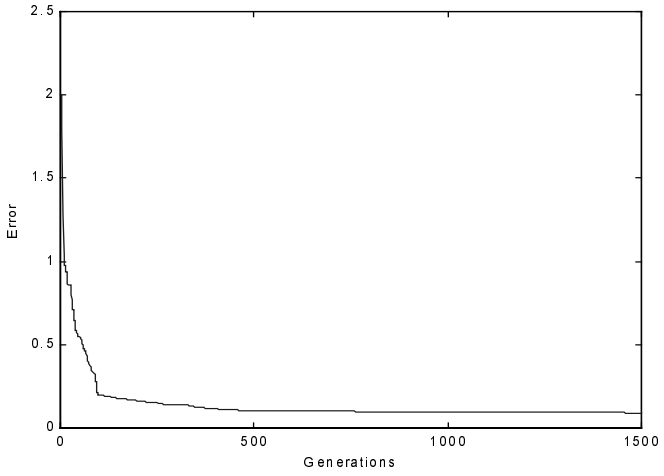

Figure 6 Performance of the genetic algorithm for $f(x_1, x_2)=s(x_1)+s(x_1)s(x_2)+s^3(x_1)s(x_2)+s(x_1)s^2(x_2)+s^3(x_2)$