Evolutionary Cellular Configurations for Designing Feed-Forward Neural Networks Architectures

G. Gutiérrez[1], P. Isasi[2], J.M. Molina[2], A. Sanchís[3], and I. M. Galván[3]

Departamento de Informática, Universidad Carlos III de Madrid,
Avenida de la Universidad 30, 28911, Leganés, Madrid.
[1]ggutierr@inf.uc3m.es,[2]{isasi,molina}@ia.uc3m.es,
[3]{masm,igalvan}@inf.uc3m.es

Abstract. In the recent years, the interest to develop automatic methods to determine appropriate architectures of feed-forward neural networks has increased. Most of the methods are based on evolutionary computation paradigms. Some of the designed methods are based on direct representations of the parameters of the network. These representations do not allow scalability, so to represent large architectures, very large structures are required. An alternative more interesting are the indirect schemes. They codify a compact representation of the neural network. In this work, an indirect constructive encoding scheme is presented. This scheme is based on cellular automata representations in order to increase the scalability of the method.

1 Introduction

The design of Neural Network (NN) architectures is crucial in the successful application of the NN because the architecture may strongly drive the neural network's information processing abilities. In most of the cases exist a large number of architectures of feed-forward neural networks set suitable to solve an approximation problem. The design of the NN architecture can be seen as a search problem within the space of architectures, where each point represents an architecture. Evidently, this search space is huge and the task of finding the simplest network that solves a given problem is a tedious and long task.
In the last years, many works have been centred toward the automatic resolution of the design of neural network architecture [1, 2, 3, 4, 5, 6]. Two main representation approaches exist to find the optimum net architecture using Genetic Algorithms (GA): one based on the complete representation of all possible connections and other based on an indirect representation of the architecture. The first one is called Direct Encoding Method, and is based on the codification of the complete network (connections matrix) into the chromosome of the GA [7, 8, 9, 10, 11]. The direct representation is relatively simple and straightforward to implement but requires much larger chromosomes. This could end in a too huge space search that could make the method impossible in practice.
In order to reduce the length of the genotype, indirect encoding scheme has been proposed in the last years. These methods consists on codifying, not the complete network, but a compact representation of it [1], avoiding the scalability problem. One

of those previous works was from Kitano [12], introducing a constructive scheme based on grammars. The solution proposed by Kitano was to encode networks as grammars. An extension of the of Kitano's method can be found in [13]. Other works have considered fractal representations [14], arguing that this kind of representation is more related with biological ideas than constructive algorithms.
In this work, an indirect constructive encoding scheme, based on Cellular Automata (CA) [15], is proposed to find automatically an appropriate feed-forward NN architecture. In this scheme, positions of several seeds in a two-dimensional grid are represented (codified) in the chromosome. The seeds are defined through two co-ordinates that indicate their positions in the grid and they are used as the initial configuration of a cellular automaton. That initial configuration is evolved using some cellular automata rules. The rules allow the convergence of the automata toward a final configuration depending on the initial configuration and it is translated into a feed-forward NN. The automata rules have been chosen such that a wide variety of feed-forward NN architecture can be obtained, from full-connected NNs to architectures with few connections.
The main interest of this paper is to show, experimentally, that cellular configurations allow obtaining appropriate architectures, as in domains relatively simple as in domains in which big architectures are required. The motivation of this approach is based on the idea that few seeds can produce big architectures. Thus, the chromosome length is reduced and the scalability of the method is increased.

2 Cellular Approach

Three different modules compose the global system proposed in this paper: the Genetic Algorithm, the Cellular Automaton and the module responsible of NN training, as is shown in Figure 1.

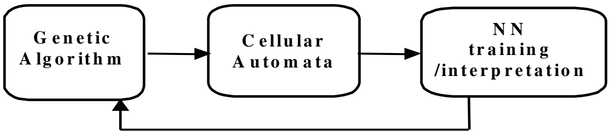

Fig.1. System's architecture and modules relationship

The GA module takes charge of generating initial configurations of the cellular automata, i.e. seed positions and to optimise these configurations from the information obtained from the training module. The cellular automaton takes the initial configuration and generates a final configuration corresponding to a particular NN architecture. Finally, the generated architecture has to be trained and evaluated for a particular problem and relevant information about the NN (error, size, etc.) is used as the fitness value for the GA. The GA carries out the architecture optimisation. However, the cellular automaton is used as a constructive way of generating the architectures. In the next subsections, the description of the different modules is presented in detail.

2.1 Genetic Algorithm Module

This module works with a population of chromosomes that codifies the seeds positions in a two-dimension grid. The GA module operates to maximise a fitness function provided by the NN module, which evaluates the efficiency of the NN to solve the considered problem.
The size of chromosomes in the GA corresponds with the number of seeds, and it codifies all the possible locations of seeds in the grid.
Chromosomes have been codified in base b, where b is the number of rows in the grid and is given through the number of inputs plus the number of outputs. Each seed is determined by a co-ordinate (x,y). A unique gene, indicating the row in which the seed is located, represents the first co-ordinate x. The second co-ordinate y will require more than one gene, if, as usual, the maximal number of hidden neurones is bigger than b. In this particular case, two genes have been used to codify the y co-ordinate, what allows a maximum of b*b hidden neurones. For instance, if there are 3 inputs and 2 outputs, the maximum size of the hidden layer is 25 (4x5+5 = 25). This could be a good estimation of the maximum number of neurones in the hidden layer, but any other consideration could be taken into account without modifying the proposed method. Hence, the chromosome will have 3 genes for each seed to be placed in the grid.

2.2 Cellular Automata Module

For generating neural networks architectures, a two-dimension CA has been used. The size of the two-dimension grid, Dimx*Dimy, is defined as follows: Dimx (rows) is equal to the number of input neurones plus the number of output neurones; Dimy (columns) corresponds with the maximum number of hidden neurones to be considered.
Each cell in the grid could be in three different states: active (occupied by a seed) or inactive. Two different kinds of seeds have been introduced: growing seeds and decreasing seeds. The first kind allows making connections and the second one removing connections. Each seed type corresponds with a different type of automata rule, so there are two rules called growing rule and decreasing rule respectively. The rules determine the evolution of the grid configuration and they have been designed allowing the reproduction of growing and decreasing seeds. In the description of the rules s is a specific growing seed, d is a decreasing seed, i is an inactive state for the cell and a means that the cell could be in any state or contains any type of seed (even a decreasing seed).
Growing rules: They reproduce a particular growing seed when there are at least three identical growing seeds in its neighbourhood. There are different configurations, growing seeds located in: rows, columns, or in a corner of the neighbourhood. In table 1(a) some of those rules are shown (the others are symmetrical). The growing rules allow obtaining feed-forward NN with a large number of connections.
Decreasing rules: They remove connections in the network deactivating a cell in the grid when the cell has a seed and a cell of its neighbourhood contains also a decreasing seed. One situation in which the decreasing rules can be applied is shown in table 1(b); the others can be obtained symmetrically.

(a)

s	s	s		s	s	s
a	i	a	⇒	a	s	a
a	a	a		a	a	a

s	s	a		s	s	a
s	i	a	⇒	s	s	a
a	a	a		a	a	a

(b)

s	d	a		s	d	a
s	s	a	⇒	s	d	a
a	a	a		a	a	a

Table 1. Some Automata Rules some configuration of seeds in the neighbourhood of a particular cell. (a) Growing rules. (b) Decreasing Rules

The mechanism of expanding the CA is as follows:

1) The growing seeds are located in the grid.
2) An expansion of the growing seeds takes place. This expansion consists on replicating each seed in turns, over its quadratic neighbourhood, in such a way that if a new seed has to be placed in a position previously occupied by another seed, the first one is replaced.
3) The growing rules are applied until no more rules could be fired.
4) The decreasing seeds are placed in the grid. If there are some other seeds in those places, they are replaced.
5) The decreasing rules are applied until the final configuration is reached.
6) The final configuration of the CA is obtained replacing the growing seeds by a 1 and the decreasing seeds or inactive cells by a 0.

2.3 Neural Network Module

To relate the final configuration of the CA with an architecture of a NN, the following meaning for a cell in the (x,y) grid is defined: if x(n, with n the number of input neurons, (x,y) represents a connection between the x-th input neuron and the y-th hidden neuron; if x>n, (x,y) represents a connection between the y-th hidden neuron and the (x-n)-th output neuron.
In the final configuration a 1 is interpreted as a connection, and a 0 as the absence of connection. Thus, the rows and columns in the matrix with values 0 are removed. A new and shorter binary matrix (M) is obtained. If Mij =1 then a connection between the i-th input neuron and the j-th hidden neuron is created, or between the j-th hidden neuron and the (i-n)-th output neuron, as is previously described. If Mij=0, there do not exist connection between that neurons.
The neural network obtained is trained to solve the particular problem considered. Weights of the NN are randomly initialised, and learned using the back propagation learning method. A value measuring the efficiency of the architecture is computed after the learning phase of the network. This value is used as the fitness function of the chromosome.

3 Experimental Results

The proposed approach is tested with two different domains, a simple one: the minimum coding problem [16], and a more complex domain: a medical classification problem, Ann-Thyroid-Database [17].
The goal is to get networks that, given an error to reach (as well as needed), its training means a lower computational effort. The meaning of computational effort is the number of weights changed along the training process. Then, neural nets with a minimal number of hidden nodes, and reaching an error as soon as possible along training, are looked for.
In the Neural Network Module, when the final matrix connection is obtained from the final configuration of CA there are some special cases take into account, following this steps:

- If there is a node in hidden layer without any connection to output, this node is eliminated from the net.
- When a hidden node has no connection from input, but it's connected to output layer, two chances have been considerate: penalizes the net and don't train it, or eliminate that node and is training.
- If an output node has no connection from hidden layer, the net is penalized and is not trained.

As it was previously mentioned, the aim is to find NN architectures requiring the least computational cost to reach an appropriate level of error. Therefore, the fitness function used in this work is defined as follows: a maximum number of learning cycles and a level of error are defined. If the network reaches that error, the training is stopped and the fitness function is evaluated as:

$$\text{Fitness} = \frac{1}{(c * tc)}; \text{c number of weights, tc training cycles} \qquad \textbf{(1)}$$

where c*tc measures the computational cost of network. When NN architectures obtained by the CA is penalized or the training of the NN end without reaching the level of error defined, maximum computational cost is assigned. Using this fitness function, the indirect encoding approach presented in this paper has been tested. The performance of this approach has been compared with direct encoding methods.

3.1 Minimum Encoding Problem

In this domain there are four inputs and two outputs. The first two inputs are just noise, and the relation between the other two inputs and the outputs is the Gray coding for integer represented by this two relevant inputs.
In this case, the direct codification is implemented as follows: the matrix of connections, previously indicated in NN Module, is codified in the chromosome. As every possible connection of a network, with 36 hidden nodes, is indicated in the chromosome, then the size of chromosome is 216. Using that codification 400 generations have been carried out and a NN with five hidden nodes is obtained (see figure 2(a)) .
In the proposed method, the position of growing and decreasing seeds in the grid of CA are codified into the genotype. The length of chromosome is 30, 3 genes for each growing or decreasing seed. Again, 400 generations are realized and the NN obtained

is shown in figure 2(b). For both approaches, the evolutions of average fitness along 400 generations are shown in figure 3. As it is possible to observe in figures 2 and 3, the indirect cellular encoding is able to provide more optimal architectures than direct encoding. 100 generations are required by the indirect encoding to find each.

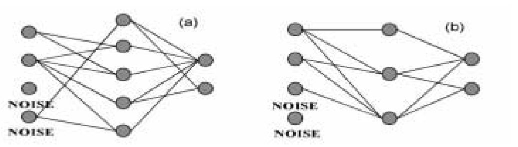

Fig. 2. Architectures obtained by direct encoding (a) and indirect encoding (b)

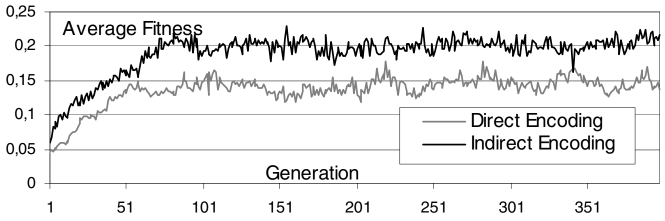

Fig. 3. Average fitness for minimum encoding problem.

3.2 Ann-Thyroid-Database

The thyroid data are measurements of the thyroid gland. Each pattern has 21 attributes, and can be assigned to any of three classes, hyper-, hypo-, and normal function of thyroid gland. This classification problem is hardly to solve for neural nets.

Experiments with a direct encoding have been done in [18]. The authors have obtained architectures with different number of hidden nodes and the connectivity percentage is calculated. One of the best architecture has 10 hidden nodes with 86 % of connectivy.

Several experiments have been developed with indirect encoding. Initially with 30 genes in the chromosome, 5 growing and 5 decreasing seeds. After, the number of seeds has been reduced to 5-3 and 3-3.Thus the length of chromosome is reduced. In all of them similar architectures have been found. Networks with 10 hidden nodes and about 72 % of connectivity are obtained. The indirect cellular encoding find smaller architectures than direct encoding. The evolution of average fitness is shown in figure 4. As it is observed in that figure few generations have been needed. With the direct encoding a large number of generation has been realized (around 1000) [18].

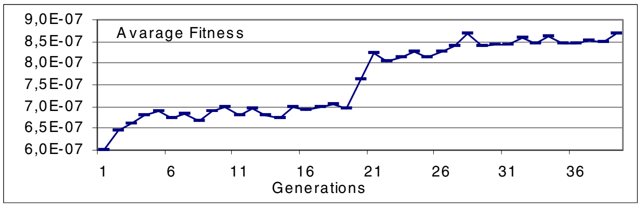

Fig. 4. Average fitness for Ann-Thyroid-Database

4 Conclusions and Future Works

The election of good neural network architectures is an important step in many problems where there is few knowledge about the problem itself. Evolutionary computation techniques are good approaches for automatically generate those good architectures. However the codification of the network is a crucial point in the success of the method. Direct codification's become inefficient from a practical point of view, making bigger and redundant the search space. To solve this problem an indirect encoding has to be used.

Indirect encoding is driven to reduce the search space in such a way that similar solutions are eliminated and represented by the only one representative. In these cases, the codification makes the method able to find better architectures.

CA are good candidates for non-direct codification's. The constructive representation introduced in this work solves some of the problems for non-direct codification's. The final representation has a reduced size and could be controlled by the number of seeds used.

The results shown that the indirect encoding approach presented in this paper is able to find appropriate NN architectures as a simple domain, as a large one. In additions the number of generations over the population is less when the indirect encoding approach is used.

Since the final configuration of the two-dimensional grid will represent a feed-forward NN connection matrix, the rules of the automata used in this first approach have been design such that a wide variety of architectures may be obtained. However, the influence of the rules in the CA evolution and the capability of the rules to generate a complete space of NN architectures must be still studied. Besides, some issues about Neural Network Module and fitness function used, i.e. how punish the nets to increase the search, will be studied in future works.

References

[1] S. Harp, Samad T. and Guha A. Towards the Genetic Synthesis of Neural Networks. Proceedings of the Third International Conference on Genetic Algorithms and their applications, pp 360-369, San Mateo, CA, USA, 1989.

[2] G.F. Miller, P.M. Todd and S.U. Hegde. Designing neural networks using genetic algorithms. In Proc. of the third international conference on genetic algorithms and their applications, pp 379-384, San Mateo, CA, USA, 1989.

[3] S. Harp, Samad T. and Guha A. Designing Application-Specific Neural Networks using the Genetic Algorithm, Advances in Neural Information Processing Systems, vol2, 447-454, 1990.

[4] F. Gruau. Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process. Proc. of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55-74, IEEE Computer Society Press, 1990.

[5] F. Gruau. Automatic Definition of Modular Neural Networks. Adaptive Behaviour, vol. 2, 3, 151-183, 1995.

[6] P.A. Castillo, J. González, J.J. Merelo, V. Rivas, G. Romero and A. Prieto. Optimization of Multilayer Perceptron Parameters using Simulated Anneling. Lectures Notes in Computer Science, Vol 1606, pp 661-670, Springer-Verlang, 1998.

[7] T. Ash. Dynamic Node Creation in Backpropagation Networks ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988), 1988.

[8] D.B. Fogel, Fogel L.J. and Porto V.W. Evolving Neural Network, Biological Cybernetics, 63, 487-493, 1990.

[9] T.P. Caudell and Dolan C.P. Parametric Connectivity: Training of Constrained Networks using Genetic Algorithms, Proc. of the third International Conference on Genetic Algorithms and their Applications, 370-374. Morgan Kaufman, 1989.

[10] J.D. Schaffer, R.A. Caruana and L.J. Eshelman. Using genetic search to exploit the emergent behaviour of neural networks. Physica D, 42, pp 244-248, 1990.

[11] E. Alba, J.F. Aldana and J.M. Troya. Fully automatic ANN design: A genetic approach. In Proc. of International workshop on artificial neural networks, pp 179-184, 1993.

[12] H. Kitano. Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476, 1990.

[13] Molina, J. M., Torresano, A., Galván, I., Isasi, P., Sanchis, A. (2000) Evolution of Context-free Grammars for Designing Optimal Neural Networks Architectures , GECCO 2000, Workshop on Evolutionary Computation in the Development of ANN. USA.

[14] J.W.L. Merril and R.F. Port. Fractally configured Neural Networks. Neural Networks, 4, 53-60, 1991.

[15] S. Wolfram. Theory and applications of cellular automata. World Scientific, Singapore, 1988.

[16] D.H. Ackley, G.E. Hinton and T.J. Sejnowski. A learning algorithm for Boltzmann machines. Cognitive Science, 9, 147-169, 1985

[17] Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[18] W. Schiffmann, M. Joost, and R. Werner. Synthesis and performance analysis of multilayer neural network architectures. Technical Report 16, University of Koblenz, Institute of Physics, 1992.