International Journal of Data Mining, Modelling and Management. 2013; 5(2): 182-191

Using genetic algorithms for automatic recurrent ANN development: an application to EEG signal classification

Daniel Rivero, Vanessa Aguiar-Pulido, Enrique Fernández-Blanco, Marcos Gestal

Fac. Informatica, University of A Coruña, Campus Elviña, 15071, A Coruña, Spain

Abstract:
ANNs are one of the most successful learning systems. For this reason, many techniques have been published that allow the obtaining of feed-forward networks. However, few works describe techniques for developing recurrent networks. This work uses a genetic algorithm for automatic recurrent ANN development. This system has been applied to solve a well-known problem: classification of EEG signals from epileptic patients. Results show the high performance of thissystem, and its ability to develop simple networks, with a low number of neurons and connections.

Keywords:
artificial neural networks; ANNs; genetic algorithms; GAs; signal classification; epilepsy detection.

1 Introduction

Artificial neural networks (ANNs) (Haykin, 1999) are systems that have returned good results in all of their application fields (Rabuñal and Dorado, 2005). Some of their typical applications include regression, classification, clustering, etc.

However, their use has some problems related to their development. This process can be divided into two parts: architecture design and training. The first part is one of the most crucial steps, because the topology of a network establishes a limit of what that network can represent. Therefore, it is important to choose the most suitable network in order to solve a particular problem. However, once a problem to be solved is known, there is no rule that says which is the best topology to solve that problem.

The second step involves setting the values of the weights of the connections. For this task, several algorithms are available, being the most used the backpropagation (BP) algorithm (Rumelhart et al., 1986).

Therefore, the development of ANNs is a process in which the human expert has to do much effort because, since the most suitable topology is unknown, he has to try different topologies, and train each one of them several times, until he finds the one that returns the best results. In order to simplify this process, several techniques have appeared that allow the developing of ANNs in a more or less automated way (Rivero et al., 2011, 2010). However, few of these works can be applied to the development of recurrent ANNs (RANNs).

RANNs are a special kind of ANNs in which recurrent connections are allowed, i.e., the output of a neuron can be used as input of itself or a previous one. In this way, the output for a single pattern depends on that pattern, but also on the outputs of the neurons computed for the previous patters. This makes this kind of network suitable for time series processing, in many different applications: filter design, forecast, etc.

This paper describes a new technique in which a genetic algorithm (GA) is used to automatically develop RANNs. With this technique, the development of ANNs becomes an automatic process in which the human expert does not have to participate. Moreover, the system described here has the advantage that it returns simplified networks, i.e., with a small number of neurons and connections.

This system was also proved with a well-known medical database. This database has a set of electroencephalograph signals from healthy and epileptic patients. The objective is to classify the signals of the database into signals showing a normal behaviour or showing seizures. The results show the high performance of the system.

2 State of the art

2.1 Genetic algorithms

A GA (Holland, 1975) is a search technique inspired in the world of Biology. More specifically, the Evolution Theory by Charles Darwin (1859) is taken as basis for its working. GAs are used to solve optimisation problems by copying the evolutionary behaviour of species. From an initial random population of solutions, this population is evolved by means of selection, mutation and crossover operators, inspired in natural evolution. By applying this set of operations, the population goes through an iterative process in which it reaches different states, each one is called generation. As a result of this process, the population is expected to reach a generation in which it contains a good solution to the problem. In GAs the solutions of the problem are codified as a string of bits or real numbers.

2.2 ANN development by means of evolutionary computation techniques

The development of ANNs has been widely treated with very different techniques in AI. The world of evolutionary algorithms is not an exception, and proof of it is the large amount of works that have been published is this aspect using several techniques (Rivero et al. 2010, 2011, 2009; Nolfi and Parisi, 2002; Cantú-Paz and Kamath, 2005). These techniques follow the general strategy of an evolutionary algorithm: an initial population with different types of genotypes encoding also different parameters – commonly, the connection weights and/or the architecture of the network and/or the learning rules – is randomly created. Such population is evaluated for determining the fitness of every individual. Subsequently, the population is repeatedly induced to evolve by means of different genetic operators (replication, crossover, mutation) until a certain termination parameter has been fulfilled (for instance, the achievement of an individual good enough or the accomplishment of a predetermined number of generations).

As a general rule, the field of ANNs generation using evolutionary algorithms is divided into three main groups: evolution of weights, architectures and learning rules.

The evolution of weight starts from an ANN with an already determined topology. In this case, the problem to be solved is the training of the connection weights, attempting to minimise the network failure. Most of training algorithms, as BP algorithm (Rumelhart et al., 1986), are based on gradient minimisation, which presents several inconveniences (Sutton, 1986). The main of these disadvantages is that, quite frequently, the algorithm gets stuck into a local minimum of the fitness function and it is unable to reach a global minimum. One of the options for overcoming this situation is the use of an evolutionary algorithm, so the training process is done by means of the evolution of the connection weights within the environment defined by both, the network architecture, and the task to be solved. In such cases, the weights can be represented either as the concatenation of binary values or of real numbers on a GA (Greenwood, 1997). The main disadvantage of this type of encoding is the permutation problem. This problem means that the order in which weights are taken at the vector might cause that equivalent networks might correspond to completely different chromosomes, making the crossover operator inefficient.

The evolution of architectures includes the generation of the topological structure. This means establishing the connectivity and the transfer function of each neuron. The network architecture is highly important for the successful application of the ANN, since the architecture has a very significant impact on the processing ability of the network. Therefore, the network design, traditionally performed by a human expert using trial and error techniques on a group of different architectures, is crucial. In order to develop ANN architectures by means of an evolutionary algorithm it is needed to choose how to encode the genotype of a given network for it to be used by the genetic operators.

At the first option, direct encoding, there is a one-to-one correspondence between every one of the genes and their subsequent phenotypes (Miller et al., 1989). The most typical encoding method consists of a matrix that represents an architecture where every element reveals the presence or absence of connection between two nodes (Alba et al., 1993). These types of encoding are generally quite simple and easy to implement. However, they also have a large amount of inconveniences as scalability (Kitano, 1990), the incapability of encoding repeated structures, or permutation (Yao and Liu, 1998).

In comparison with direct encoding, there are some indirect encoding methods. In these methods, only some characteristics of the architecture are encoded in the chromosome and. These methods have several types of representation.

Firstly, the parametric representations represent the network as a group of parameters such as number of hidden layers, number of nodes for each layer, number of connections between two layers, etc. (Harp et al., 1989). Although the parametric representation can reduce the length of the chromosome, the evolutionary algorithm performs the search within a restricted area in the search space representing all the possible architectures. Another non-direct representation type is based on a representation system that uses the grammatical rules (Kitano, 1990). In this system, the network is represented by a group of rules, shaped as production rules that make a matrix that represents the network, which has several restrictions.

The growing methods represent another type of encoding. In this case, the genotype does not encode a network directly, but it contains a group of instructions for building up the phenotype. The genotype decoding will consist on the execution of those instructions (Nolfi and Parisi, 2002), which can include neuronal migrations, neuronal duplication or transformation, and neuronal differentiation.

Another important non-direct codification is based on the use of fractal subsets of a map. According to Merrill, the fractal representation of the architectures is biologically more plausible than a representation with the shape of rules (Merrill and Port, 1991). Three parameters are used which take real values to specify each node in an architecture: a border code, an entry coefficient and an exit code.

One important characteristic to bear in mind is that all those methods evolve architectures, either alone (most commonly) or together with the weights. The transfer function for every node of the architecture is supposed to have been previously fixed by a human expert and is the same for all the nodes of the network – or at least, all the nodes of the same layer–, despite of the fact that such function has proved to have a significant impact on network performance (DasGupta and Schnitger, 1992). Only few methods that also induce the evolution of the transfer function have been developed (Hwang et al., 1997).

With regards to the evolution of the learning rule, there are several approaches (Turney et al. 1996), although most of them are only based on how learning can modify or guide the evolution and also on the relationship among the architecture and the connection weights. There are only few works focused on the evolution of the learning rule itself.

3 Model

The model proposed in this paper allows the obtaining of RANNs with no limitation in their architecture. Any kind of connectivity is allowed. In order to obtain this, a GA is used to evolve the networks. Each individual of the GA represents a specific topology with the connection weights already set, i.e., that network can be evaluated with the patterns set. Therefore, the chromosome must represent all of this information, including weights and connectivity.

Each neuron of a recurrent network with M inputs, N hidden neurons and P outputs can receive inputs from any other neurons. Therefore, each neuron can have up to $M + N + P$ input connections. The only exception is the input neurons, which do not receive inputs from other neurons. So, a recurrent network can have up to $(N + P) * (M + N + P)$ connections, and the chromosome will need this length in order to evolve the weights of the connections. However, it is possible not to have totally connected networks and not use some of these connections. For this task, each connection will have two values in the genotype: a floating point value, that stores the weight, and a Boolean value, that says if that connection is present or not.

The chromosome also allows the possibility that some of the hidden neurons are not used. In order to codify this, N additional Boolean values are stored in the chromosome, each one for each neuron, that specify if that neuron is used or not. If a specific neuron is not used, all of the input and output connections are deleted as if their corresponding bits in the chromosome specified that these connections are not used.

Therefore, the length of the chromosome is $(N + P) * (M + N + P) * 2 + N$, including floating point and Boolean values. This chromosome encodes a network topology and weight configuration. Therefore, the network is ready for being evaluated. However, before this is done, the networks undergo a simplification process, because it is possible that a network has some parts not used. This is the case of neurons that do not have input or output connections. In this case, these neurons can be deleted. This is done in an iterative process in which, in each step, the network is examined for these kinds of neurons. If a neuron is found with no input or output connections, this neuron is eliminated along with its remaining input or output connections. This process is repeated until no neurons are eliminated. Figure 1 shows this iterative process to simplify three different networks.

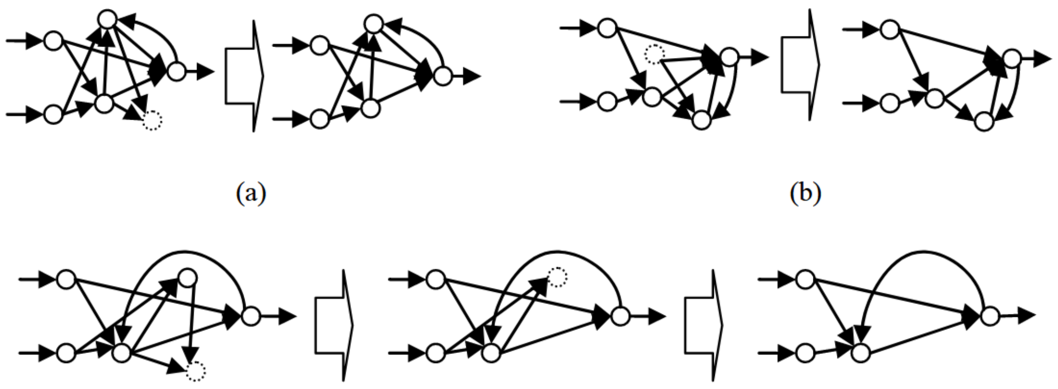

Figure 1. Simplification of the networks

The evaluation of an individual includes the building of a network, simplifying its topology (by deleting useless neurons and connections), setting the values of the connection weights and evaluating this network with the pattern set. The error value obtained as result of this evaluation with the pattern set is returned as the fitness of the individual.

Nevertheless, this error value considered as fitness value has been modified in order to induce the system to generate simple networks. The modification has been made by adding a penalisation value multiplied by the number of connections of the network, because this number is the determining factor of the network’s complexity. In such way, and given that the evolutionary system has been designed in order to minimise an error value, when adding a fitness value, a larger network would have a worse fitness value. Therefore, the existence of simple networks would be preferred as the penalisation value that is added is proportional to the number of connections at the ANN. The calculus of the final fitness is as follows:

$$fitness = MSE + N * P$$

where *MSE* is the mean square error of the ANN within the group of training patterns, *N* is the number of connections of the network and *P* is the penalisation value for such number of connections.

4 Description of the problem

The system described in this work has been used in order to solve a well-known problem: classification of EEG signals. The problem to be solved is to classify seizures in EEG signals taken from epileptic patients. Epilepsy is one of the most common neurological disorders, and is characterised by the occurrence of recurrent seizures in the EEG signal (Mohseni et al. 2006). This signal measures the brain electrical activity, and its analysis is one of the most important tools for the diagnosis of neurological disorders. The recording of EEG signals by means of the use of recording systems generate large amounts of EEG data, and the complete visual analysis of this data by an expert is not routinely possible. This is the reason why efforts have long been made in order to develop tools that automatically process EEG signals.

The data used in this paper is well described by Andrzejak et al. (2001), and is publicly available. The complete dataset consists of five sets (denoted A–D), each containing 100 single-channels EEG signals of 23.6 s., with a sampling rate of 173.61 Hz. These segments were selected and cut out from continuous multichannel EEG recordings after visual inspection for artefacts, e.g., due to muscle activity or eye movements. In addition, the segments had to fulfil a stationarity criterion. Sets A and B consisted of segments taken from surface EEG recordings that were carried out on five healthy volunteers using a standardised electrode placement scheme. Volunteers were relaxed in an awake state with eyes open (set A) and eyes closed (set B), respectively. Sets C, D, and E originated from an EEG archive of presurgical diagnosis. Segments in set D were recorded from within the epileptogenic zone, and those in set C from the hippocampal formation of the opposite hemisphere of the brain. While sets C and D contained only activity measured during seizure free intervals, set E only contained seizure activity. Here, segments were selected from all recording sites exhibiting ictal activity.

The present study aims to do the classification between segments with seizures from epileptic patients, and normal intervals, from healthy volunteers.

5 Results

The system described in the paper has been applied to solve this particular problem. 200 different segments (100 for each class, normal and seizure) were used. Each segment has 4,097 samples (23.6 seconds with a sampling rate of 173.61 Hz).

Each segment was windowed with a window size of 256 and an overlapping of 64, which lead to having 21 different windows. From each window, 6 simple time and frequency-based features were extracted:

- X_0: mean of the window
- X_1: standard deviation of the window
- X_2: mean of the absolute values of the FFT of the window
- X_3: standard deviation of the absolute values of the FFT of the window
- X_4: mean of the power spectral densities of the values of the window
- X_5: standard deviation of the power spectral densities of the window.

Therefore, each segment has 6 * 21 = 126 features. However, these features are inserted consecutively to the RANN. The RANN has only 6 inputs, corresponding to the features extracted from each segment. After the features of a window are used as inputs, the network is evaluated and the inputs from the following window are used. However, the error is not evaluated on each window, but only one error value is calculated for the whole segment.

Before evaluating the first window a segment, the RANN is reset, to delete all of the values of the previous computations of the neurons. Then, the 6 features of each of the 21 windows are consequently presented as inputs to the RANN. As result, the network generates 21 outputs. The final result for that segment is the average value of those 21 outputs, and this is the result used for the calculus of the error in that segment.

The 200 different segments were randomly divided into two non-overlapping groups, for training (50%) and test (50%).

Some preliminary experiments were performed in order to tune the parameters of the system. According to those results, the system was run with the following parameters, which have shown to give good results:

- population size: 1000 individuals
- crossover rate: 90%
- mutation probability: 3%
- selection algorithm: two-individual tournament
- crossover algorithm: two-point crossover
- penalisation to the number of connections: 0.00001
- the system was run allowing the finding of networks with a maximum of 20 hidden neurons.

With this configuration set, different networks were obtained that solved this problem. 200 independent executions were run. The results presented on this paper are the average value of the results obtained in those executions, shown on Table 1. This table shows the average value of the mean square error and mean error obtained in the training and test sets. This table also shows the average values of the number of hidden neurons and connections of the best networks found in each run.

Table 1. Results obtained

		Training	Test
MSE	Average	0.0048	0.0598
	Std.	0.0115	0.0301
ME	Average	0.0141	0.0598
	Std.	0.0133	0.0301
Number of neurons		6.95	
Number of connections		64.63	

As can be seen in this table, very high accuracies can be obtained with this system. A mean error of 0.0141 means having an accuracy of 98.59%. However, one of the most interesting features of this system is the low number of neurons and connections. The values shown on Table 1 are the average number of neurons and connections of the 200 independent runs. The system was run allowing the obtaining a maximum of 20 hidden neurons (which lead having a maximum of 567 connections). However, the networks found have a much lower number of neurons and connections. This means that, even setting this initial number of hidden neurons very high, the system can obtain simple networks.

6 Conclusions

This paper describes a system that allows the automatic obtaining of RANNs with no participation of the expert. As results show, these networks found can lead to having very good results. In this paper, a classification problem was researched. However, this system could be applied to other type of problems in signal processing: prediction, filtering, etc.

One of the most interesting features of this system is that it allows the obtaining of simple networks, with a low number of neurons and connections. Even the initial number of hidden neurons was set too high (20), the system was able to find networks with a much lower number of neurons and connections.

7 Future works

From this work, new research lines can be taken. First, the parameters involved in the configuration of the system have to be researched, because they are believed to have a great impact in the performance of the system.

With respect of the application, new features can be extracted in order to further improve the results. Also, this system could also be proved with other type of problems involving signals, such as prediction or signal filtering.

Acknowledgements
This work is supported by the 'Galician Network for Colorectal Cancer Research' (REGICC, ref. 2009/58), from the General Directorate of Scientific and Technologic Promotion of the Galician University System of Xunta de Galicia, by the 'Ibero-American Network of the Nano-Bio-Info-Cogno Convergent Technologies', Ibero-NBIC Network (209RT0366) funded by CYTED (Ciencia y Tecnología para el Desarrollo), 'PHR2.0: Registro Personal de Salud en Web 2.0' (ref. TSI-020110-2009-53) funded by the Spanish Ministry of Industry, Tourism and Trade and the grant, 'Development of new image analysis techniques in 2D Gel for biomedical research' (ref. 10SIN105004PR) funded by Xunta de Galicia, (ref. PIO52048) and RD07/0067/0005, funded by the Carlos III Health Institute. The development of the experiments described in this work, has been performed with equipments belonging to the Super Computation Center of Galicia (CESGA).

References

Alba, E., Aldana, J.F. and Troya, J.M. (1993) 'Fully automatic ANN design: a genetic approach', *Lecture Notes in Computer Science, Proc. Int. Workshop Artificial Neural Networks (IWANN'93)*, Vol. 686, pp.399–404.

Andrzejak, R.G., Lehnertz, K., Rieke, C., Mormann, F., David, P. and Elger, C.E. (2001) 'Indications of non-linear deterministic and finite dimensional structures in time series of brain electrical activity: dependence on recording region and brain state', *Physical Review E*, Vol. 64, No. 6, 0619071–8.

Cantú-Paz, E. and Kamath, C. (2005) 'An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems', *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics*, Vol. 35, No. 5, pp.915–927.

Darwin, C.R. (1859) *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, 1st ed., 1st issue, John Murray, London.

DasGupta, B. and Schnitger, G. (1992) 'Efficient approximation with neural networks: a comparison of gate functions', Dep. Comput. Sci., Pennsylvania State Univ.

Greenwood, G.W. (1997) 'Training partially recurrent neural networks using evolutionary strategies', *IEEE Transactions on Speech Audio Processing*, Vol. 5, No. 2, pp.192–194.

Harp, S.A., Samad, T. and Guha, A. (1989) 'Towards the genetic synthesis of neural networks', in Schafer, J.D. (Ed.): I*nt. Conf. Genetic Algorithms and Their Application*s, Morgan Kaufmann, San Mateo, CA.

Haykin, S. (1999) *Neural Networks*, 2nd ed., Prentice Hall, Englewood Cliffs, NJ.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

Hwang, M.W., Choi, J.Y. and Park, J. (1997) 'Evolutionary projection neural networks', *IEEE Int. Conf. Evolutionary Computation*.

Kitano, H. (1990) 'Designing neural networks using genetic algorithms with graph generation system', *Complex Systems*, Vol. 4, pp.461–476.

Merrill, J.W.L. and Port, R.F. (1991) 'Fractally configured neural networks', *Neural Networks*, Vol. 4, No. 1, pp.53–60.

Miller, G.F., Todd, P.M. and Hedge, S.U. (1989) 'Designing neural networks using genetic algorithms', in Kaufmann, M. (Ed.): *Third International Conference on Genetic Algorithms*, San Mateo, CA.

Mohseni, H.R., Maghsoudi, A. and Shamsollahi, B. (2006) 'Seizure detection in EEG signals: a comparison of different approaches', *Engineering in Medicine and Biology Society, EMBS06*, 28th Annual International Conference of the IEEE.

Nolfi, S. and Parisi, D. (2002) 'Evolution of artificial neural networks', *Handbook of Brain Theory and Neural Networks*, 2nd ed., MIT Press, Cambridge, MA.

Rabunal, J.R. and Dorado, J. (2005) *Artificial Neural Networks in Real-Life Applications*, Idea Group, London.

Rivero, D., Dorado, J., Rabuñal, J.R. and Pazos, A. (2011) 'Database analysis with ANNs by means of graph evolution', in Zhang, Q., Segall, R.S. and Cao, M. (Eds.): *Visual Analytics and Interactive Technologies: Data, Text and Web Mining Applications*, Information Science Reference.

Rivero, D., Dorado, J.R., Rabuñal, J. and Pazos, A. (2009) 'Evolving simple feed-forward and recurrent ANNs for signal classification: a comparison', *Neural Networks, IEEE – INNS – ENNS International Joint Conference on*.

Rivero, D., Dorado, J.R., Rabuñal, J. and Pazos, A. (2010) 'Generation and simplification of artificial neural networks by means of genetic programming', *Neurocomputing*, Vol. 73, Nos. 16–18, pp.3200–3223.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) 'Learning internal representations by error propagation', *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1, pp.318–362.

Sutton, R.S. (1986) 'Two problems with backpropagation and other steepest-descent learning procedure for networks', *8th Annual Conference on Cognitive Science Society*, Erlbaum, Hillsdale, NJ.

Turney, P., Whitley, D. and Anderson, R. (1996)'Special issue on the Baldwinian effect', *Evolutionary Computation*, Vol. 4, No. 3, pp.213–329.

Yao, X. and Liu, Y. (1998) 'Toward designing artificial neural networks by evolution', *Applied Mathematical Computatio*n, Vol. 91, No. 1, pp.83–90.