Artificial Neural Networks Generation Using Grammatical Evolution

Khabat Soltanian[1], Fardin Akhlaghian Tab[2], Fardin Ahmadi Zar[3], Ioannis Tsoulos[4]
[1]Department of Software Engineering, University of Kurdistan, Sanandaj, Iran, k.soltanian@uok.ac.ir
[2]Department of Engineering, University of Kurdistan, Sanandaj, Iran, f.akhlaghian@uok.ac.ir
[3]Department of Engineering, University of Kurdistan, Sanandaj, Iran, f.ahmadizar@uok.ac.ir
[4]Department of Computer Science, University of Ioannina, Ioannina, Greece, itsoulos@cs.uoi.gr

Abstract: *in this paper an automatic artificial neural network generation method is described and evaluated. The proposed method generates the architecture of the network by means of grammatical evolution and uses back propagation algorithm for training it. In order to evaluate the performance of the method, a comparison is made against five other methods using a series of classification benchmarks. In the most cases it shows the superiority to the compared methods. In addition to the good experimental results, the ease of use is another advantage of the method since it works with no need of experts.*

Keywords- artificial neural networks, evolutionary computing, grammatical evolution, classification problems.

1. Introduction

Artificial Neural Networks (ANNs) have risen from 1960s as a novel way to mimic the human brain [1]. The learning ability of ANN makes it a powerful tool for various applications such as pattern recognition, classification, clustering, vision, control systems, and prediction. Designing network architecture and training it are the most important problems with the exploiting ANNs. The Back Propagation (BP) method [1] is the most known method for the training phase of the neural networks and is used widely in a variety of tasks. Selecting effective input features, determining the number of hidden neurons, and the connectivity pattern of neurons is the task of ANN topology designing and usually needs to be performed by experts.

Evolutionary Neural Networks refers to a class of research that Evolutionary Algorithms (EAs) are used in ANN designing or training or in the both. Evolutionary Algorithms are a group of population based stochastic search algorithms that simulate the natural evolution. As evolutionary algorithms search globally (are less likely to be trapped in local minima than traditional methods) and can handle infinitely large, non-differentiable and multimodal search space of architectures, this field has been interested by many researchers and ANN developers [2].

In this paper a method is presented that uses GE for designing ANN topology and uses BP for training it (GE-BP). GE is previously used for neural networks construction and training and successfully tested on multiple classification and regression benchmarks [3]. In [3] the grammatical evolution creates the architecture of the neural networks and also suggest a set for the weights of the constructed neural network. Whereas the grammatical evolution is established for automatic programming tasks not the real number optimization. Thus, we altered the grammar to generate only the topology of the network and used BP algorithm for training it.

The following section reviews several evolutionary neural networks approaches that have been appeared in the literature previously. The section III describes the grammatical evolution algorithm and settings that is used in the proposed method. Details of using GE in neural networks topology design and are presented in Section IV. Section V contains the experiments that have been performed and results that acquired to evaluate our method against other evolutionary neural network methods and. Section VI concludes the paper.

2. Literature Review

Evolutionary neural networks can be divided in three major groups. The first is the adaptation of neural network weights by evolutionary algorithms that is a form of neural network training. The second group of works is the use of EAs for designing the architecture of ANN and the third is the effort that was performed for simultaneous evolution of weights and architecture [2].

In the evolutionary training of ANNs, the architecture of network is determined before training, usually by experts. As the training of ANNs can be formulated as a search problem, the evolutionary algorithm is used for exploring the search space for finding an optimized set of weights. Binary representation of connection weights is one of the earlier works in this area [4]. Also some of researchers used the real vector for representing connection weights and subsequently used suitable real encoding evolutionary algorithms [5-6]. Since ES and EP

978-1-4673-5634-3/13/$31.00 ©2013 IEEE

algorithms are well-suited for real vector optimization, they are used in this area in many cases [2, 7].

The architecture of the network has great importance because it affects the learning capacity and generalization capability of the neural network [6]. The task of ANN design is usually done by a human expert who should try many structures in order to find an optimized one. The hill-climbing approaches such as constructive and destructive algorithms are used for automated design of the architecture of networks [8-9]. But, the main drawback of these methods is that they are quite susceptible to fall in a local optimum [10].

In the evolution of architecture, there are two approaches for representing the chromosome. In the first approach, named direct encoding, all of the network architecture details are coded in the chromosome. Assuming the neural network as a graph and using an adjacent matrix for representing its genotype is one of the direct encoding methods. In this method, each entry is a binary number that shows the presence or the absence of a connection between two nodes [4, 11].

Other methods of representation are indirect encodings, in which only some properties of architecture are encoded in the chromosome. The existing knowledge of destination architecture can be used to reduce the search space by determining some properties of the architecture. For example if we know that full connection architecture is suitable for our problem, it is sufficient to encode only the number of hidden layers and the number of neurons in each layer in the chromosome [12]. Kitano introduced a grammar based indirect representation that encodes production rules in the chromosome instead of adjacent matrix of the network. Compact genotype is the main advantage of this method. However, it has some drawbacks [2]. Fractal representation inspired by regularity, symmetry and self-similarity of live organisms is another indirect encoding schema that may be more plausible than other encoding [13] Gruau has introduced Cellular encoding (CE) as an indirect representation that is motivated by cell division in nature [14].

NeuroEvolution of Augmenting Topologies (NEAT) [16], presented by Stanley and Miikkulainen is another successful system in this area. Grammatical Evolution [17] is used by Tsoulos and et al [3] for construction and training ANN with one hidden layer and evaluated on multiple classification and regression problems. As one of the recent works, Rivero and et al [18] used GP for designing and training a feed forward ANN with any arbitrary architecture.

In this paper a method is presented that uses Grammatical Evolution for designing ANN topology and uses BP for training it. In the proposed method, only the architecture of neural network is encoded in the chromosome. In the other words, the number of hidden neurons and the connectivity of neurons are determined by GE and the weights are determined by BP algorithm.
In [3] the weights of connections are evolved along with network topology and the GE is performed to optimize them whereas the generation of a number passes through multiple steps in GE and optimization of real numbers is not straightforward. So, we altered the algorithm to generate the architecture of the network while the weights of generated network are optimized using BP algorithm.

3. Proposed Method

The Grammatical Evolution introduced by Ryan and O'Neill [17] and has been successfully used for various problems such as symbolic regression, robot control, financial forecasting, feature selection, and NN training and construction [3]. Grammatical evolution is the combination of a CFG and a genetic algorithm. The CFG grammar is used for genotype decoding and a genetic algorithm is used to evolve chromosomes.

3.1 ANN Architecture Generation Using the Proposed CFG

The architecture of a feed forward neural network that has one hidden layer can be represented as a weighted summation of units, whereas each unit is a sigmoid function of summation of weighted inputs. Fig. 1. shows a neural network with two hidden neurons, four input signals, and one output neuron, as the first neuron is connected to 1st and 3rd inputs and the second neuron is connected to 2nd and 4th inputs. The string representation of this neural network can be expressed as: "w × sigmoid (w × x1 + w × x3 + biasW1) + w × sigmoid (w × x2 + x × x4 + biasW2)", while 'w' denotes the weight of connections and 'sig' represents the sigmoid function of neurons. In this paper a BNF grammar is used to generate the ANN architecture. The value of each weight is determined by BP algorithm.

A BNF grammar is denoted as G (N, T, S, P), as N is a set of non-terminal symbols, T is a set of terminal symbols, S is a non-terminal symbol that starts the genome decoding process and P is a set of production rules in the form $A \rightarrow a$ or $A \rightarrow aB$, as 'A' and 'B' are non-terminal symbols and 'a' is a terminal symbol. Terminal symbols are those that can be present in a valid

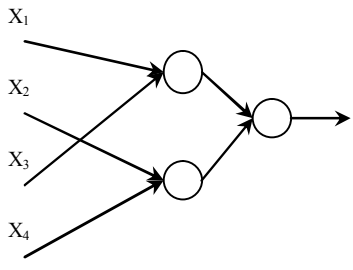

Fig. 1: A feed forward neural network.

```
<S>       ::= <Node> | <S> + <Node>
<Node>    ::= w * sig(<Sum> + w)
<Sum>     ::= w * <xList> | <Sum> + w * <xList>
<xList>   ::= x1 | x2 | x3 | x4
```

Fig. 2: The proposed CFG grammar.

sentence of the grammar and non-terminals are symbols that participate in decoding process as temporary symbols.

Fig. 2. shows the BNF grammar that is proposed in this paper for generating the ANN architectures. In this grammar the terminal symbols are: {+, -, *, sig(), x1, x2, x3, x4, 0, 2, 3, 4, 5, 6, 7, 8, 9} and the non-terminal symbols are: {<S>, <Node>, <Sum>, <xList>}. A sentence is called valid sentence if it has not any non-terminal symbols.

Fig. 3. shows a mapping example in which genotype decoding begins with start symbol (S); as it is a non-terminal symbol, it must be altered with one of the successors of rules 1 or 2 (<Node> or <S> + <Node>). The first gene in the chromosome (27) determines the production rule that must be replaced with <S>; the reminder of division the gene value divide to the number of production rules conclude the index of production rule that must be fired. As there is two production rules (with predecessor <S>) and the 27 mod 2 equals to 1, the production rule which its successor must be replaced with <S> is: <S> + <Node>. All non-terminal symbols must be replaced from left in the produced string in this way.

3.2 Used Genetic Algorithm

The pseudo-code of genetic algorithm used is shown in Fig. 4. In the initialization, each chromosome is created as an array of random integer numbers in range [0, 255]. In the described method the size of each chromosome is 100. With inspiration from selection pressure of environment that determines the chance of each living individual to mate and survive in the nature, the selection chance of each individual in the population to be parent or be moved to the next generation is based on its fitness against its competitors. In this paper we used rank-based roulette wheel [19] method for both parent and survivor selection. In survivor selection, we used elitism for protecting good individuals from disappearing of the population; before the roulette wheel selection is performed, a 10% of the population is moved to the next generation intact.

In the described method, the single point crossover is used for producing two new individuals from two parents. The mutation in this approach is simple too, as each gene with a small probability (mutation rate) may be altered with a random number in rage [0,255]. The termination time of algorithm is controlled by the number of generations.

3.3 Fitness Evaluation

The architecture of each ANN is generated through grammatical evolution process and subsequently the produced ANN is trained using the BP algorithm. For training procedure, the training examples $(X_i, y_i), 1 \leq i \leq m$ are used, where X_i, denotes the input vector of example i, y_i stands for the real output of the vector and M is the total number of the training examples. The mean squared error (MSE) of the ANN after the training is used as fitness value of the individuals and is calculated as:

$$f = \frac{\sum_{i=1}^{m}(O(X_i) - y_i)^2}{m}, \quad (1)$$

Where $O(X_i)$ is the output of trained ANN for example i.

4. Experiment Results

The proposed method is compared against other design and training procedures in this section. All methods have been tested on a series of the classification problems. Table I. shows a summary of the used classification datasets after normalization.

In [6] the normalization method is described in detail. The number of classes, examples and input features of the dataset and also the number of training epochs (in BP algorithm) of the designed ANN for each dataset has been mentioned in this table. In comparing classification algorithms, the k-fold cross validation is the most common method [18]. In k-fold cross validation method the dataset D is divided in to k non-overlapping sets $D_1, D_2, \ldots, D_k$. The algorithm runs k times. In the iteration i, the dataset $D \backslash D_i$ is used as the train set and D_i is used as the test set of the algorithm. However, we used 2-fold cross validation in our experiments. In the other words, in each execution, the total dataset is divided randomly to two halves, and 2-fold cross validation is performed. This method is used for evaluating the evolutionary neural network algorithms, which the proposed method is compared with them; whereas the 2-

Chromosome: 27, 8, 13, 6, 12, 2, 5, 0, 1, 3

Generated string	Gene in use	Production number
<S>	27	1
<S>+<Node>	8	0
<Node>+<Node>	non	
w*sig(<Sum>+w)+<Node>	13	1
w*sig(<Sum>+w*<xList>+w)+<Node>	6	0
w*sig(w*<xList>+w*<xList>+w)+<Node>	12	0
w*sig(w*x1+w*<xList>+w)+<Node>	2	2
w*sig(w*x1+w*x3+w)+<Node>	non	
w*sig(w*x1+w*x3+w)+w*sig(<Sum>+w)	5	1
w*sig(w*x1+w*x3+w)+w*sig(<Sum>+w*<xList>+w)	0	0
w*sig(w*x1+w*x3+w)+w*sig(w*<xList>+w*<xList>+w)	1	
w*sig(w*x1+w*x3+w)+w*sig(w*x2+w*<xList>+w)	3	
w*sig(w*x1+w*x3+w)+w*sig(w*x2+w*x4+w)		

Fig. 3: Generating neural network topology of Fig. 1. through the CFG of Fig. 2.

fold cross validation is performed 5 times (5×2 cross-validation) and we used this method to do experiments and compare our method with them.

TABLE I. The Description Of The Used Datasets.

Dataset	classes	examples	Input features	BP epochs
Breast cancer	2	699	9	20
Heart-Cleveland	2	303	21	40
Ionosphere	2	351	34	40
Iris	3	150	4	80

The arithmetic mean of accuracies of the algorithm on the test datasets in the 10 different tests (5 executions, and 2 tests in each execution) is the main measure for comparing the method proposed here with other methods. Also, the standard deviation of the algorithm accuracies of these 10 tests is calculated and compared with the competitor techniques.

1. Initialize population (create random chromosomes): G(0) and set i=0;
2. REPEATE
   a. Map each individual in the population to the its phenotype
   b. Evaluate them
   c. Select parent from G(i) based on their fitness in the population
   d. Apply crossover and mutation to parents and generate offspring
   e. Select individuals from G(i) and offspring based on their fitness to participate in G(i+1)
   f. i=i+1
3. UNTIL Termination criterion is satisfied

Fig. 4: Pseudo-code of the genetic algorithm.

In this paper the number of fitness evaluation is the basis of comparing the complexity of algorithms. So, the number of generations multiplies by the size of population is calculated as the complexity of the algorithm.

The parameters of an evolutionary algorithm affect its performance. The crossover and mutation rates, the individual's chance of selection and the number of generations are parameters that must be determined in our proposed algorithm. Table II. Shows the parameter values employed in this algorithm, to perform comparisons.

Table III. and IV shows the accuracy, the standard deviation, and the complexity of the proposed method and other methods on the classification datasets. These results have been acquired from 5×2 cross validation test. The evolutionary neural network methods with which comparisons performed can be categorized in two categories: techniques which use EA to design the architecture of ANNs and techniques that design and train ANNs simultaneously by means of EAs.

The Matrix, Parameters, and Grammar methods use EAs to design the network while determine the weights of the connections by BP. Details of the implementation of these methods are described in [6] and the results in table III are obtained in [6]. GE [3] and Modified-ES [21] can be categorized as second group of method, since they evolve both architectures and weights. The GE was re-implemented by the authors and the results have been acquired in same environment that was explained in origin paper but in 5×2 cross validation tests. In Modified-ES, the authors proposed a method for architecture and weights adaptation by adding a new mutation (connection pruning) to the evolution strategy algorithm. The results of our method described here are reported in table IV under column named GE-BP.

TABLE II. Parameter Setting Of The Algorithm.

Population size	100
Crossover probability	0.9
Mutation probability	0.05
Parent selection parameter (S)	1.99
Survivor selection parameter (S)	1.8
Elites	Pop_size/10

For each method in Table III-IV, there are three columns that correspond from left to right to accuracy mean, accuracy standard deviation and the complexity of algorithms.

As indicated by the tables, the results obtained by the proposed method are in the same order as other methods and in most cases the proposed method overcomes them.

In addition to the good results, our proposed method has other features that make it more powerful method. Independency to the experts is one of the major features of the proposed method, while some other methods require the expert's participations. For example, Parameter and Grammar methods still need expert efforts to determine the connectivity pattern of destination network and in the Matrix and Modified-ES methods the maximum number of hidden neurons must be determined. In the contrary to other methods (except for GE) our system has the ability to use the expert knowledge in exploring the search space.

If we know that some features have more influence on network performance, we can increase its selection chance by modifying the CFG grammar (repeating it in the <xList> rules). For example, in the Fig. 2 the selection probability of each feature is equal to 0.25, while if we modify the rule as following:

```
<xList> ::= x1 | x2 | x3 | x4 | x4,
```

The selection probability of x4 will be 0.4 and selection probability of each of other features would be 0.2.

TABLE III. Means And Standard Deviations Of Accuracies Obtained From Architecture Evolution Methods And Their Computational Complexities..

Dataset	Matrix			Parameter			Grammar		
Breast cancer	**96.77**[a]	1.10[b]	92000[c]	96.69	1.13	100000	96.71	1.16	300000
Heart-Cleveland	76.78	7.87	304000	65.89	13.5	200000	72.8	12.6	600000
Ionosphere	87.06	2.14	464000	85.58	3.08	200000	88.03	1.55	600000
Iris	92.40	2.67	320000	91.73	8.26	400000	92.93	3.08	1200000

[a] accuracy mean [b] accuracy standard deviation [c] computational complexity

TABLE IV. Means And Standard Deviations Of Accuracies Obtained From Our Method, GE, and Modified-ES Methods And Their Computational Complexities.

Dataset	GE			Modified-ES			GE-BP		
Breast cancer	95.61	0.93	250000	96.28	2.54	70000	95.90	3.14	92000
Heart-Cleveland	80.20	1.52	250000	**82.91**	8.62	70000	80.20	5.24	200000
Ionosphere	89.63	2.19	250000	89.12	2.54	70000	**89.90**	3.16	200000
Iris	94.80	1.26	250000	96.53	3.17	70000	**96.6**	6.14	250000

5. Conclusion

In this paper a method for automatic neural network designing by means of grammatical evolution is presented; while the BP algorithm is used for its training. The system is compared with 5 evolutionary neural network methods on well known classification datasets. In the same computational complexity, the comparisons are performed and our method shows superiority in most cases. In addition, independency to the experts and the ability of utilizing the expert knowledge for more efficient space exploration are the major features of the method described here.

Another interesting research line that is followed by the authors is the using a globally search algorithm to adapt the connection weights instead of the BP algorithm that searches locally. Also, modifying the grammar for generating recurrent neural networks and evaluating it on some time series can be followed as a research topic.

References

[1] B. Widrow and M.A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proceedings of the IEEE,* vol. 78, pp. 1415-1442, 1990.

[2] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE,* vol. 87, pp. 1423-1447, 1999.

[3] I. Tsoulos, D. Gavrilis, E. Glavas, "Neural Network Construction and Training using Grammatical Evolution," *Natural Computing,* vol. 72, pp. 269-277, 2008.

[4] D. Whitley, "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, 1989, pp. 116-123.

[5] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," 1989.

[6] E. Cantu-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems," *Systems, Man and Cybernetics, Part B, IEEE Transactions on,* vol. 35, pp. 915-927, 2005.

[7] D. B. Fogel, E. C. Wasson, E. M. Boughton, "Evolving neural networks for detecting breast cancer," *Cancer letters,* vol. 96, pp. 49-53, 1995.

[8] M. Frean, "The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks," *Neural Computation,* vol. 2, pp. 198-209, 1990/06/01 1990.

[9] P. Bartlett*, et al.*, *Training a neural network with a genetic algorithm*: University of Queensland, 1990.

[10] P. Angeline, G. Saunders, J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Transactions on Neural Networks,* vol. 5, pp. 54-65, 1994.

[11] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems Journal,* vol. 4, pp. 461-476, 1990.

[12] S. Harp, T. Samad, A. Guha, "Toward the Genetic Synthesis of Neural Networks," in *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[13] F. Gruau, "Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm," 1994.

[14] F. Gruau, "Genetic synthesis of modular neural networks," 1993, pp. 318-325.

[15] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *Neural Networks, IEEE Transactions on,* vol. 8, pp. 694-713, 1997.

[16] K. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation,* vol. 10, pp. 99-127, 2002.

[17] C. Ryan, JJ. Collins, M. Neill, "Grammatical evolution: Evolving programs for an arbitrary language," *Genetic Programming,* pp. 83-96, 1998.

[18] D. Rivero, J. Dorado, J. Rabunal, A. Pazos, "Generation and simplification of Artificial Neural Networks by means of Genetic Programming," *Neurocomputing,* vol. 73, pp. 3200-3223, 2010.

[19] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*: Springer Verlag, 2003.

[20] T. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation,* vol. 10, pp. 1895-1923, 1998.

[21] K. Soltanian, F. Ahmadizar, F. Akhlaghian Tab, "Artificial Neural Networks design and training using evolution strategy," presented at the First CSUT Conference on Computer, Communication, Information Technology, Tabriz, Iran, 2011.