2005 IEEE International
Symposium on Signal Processing
and Information Technology

Neural Network Construction using Grammatical Evolution

Ioannis G. Tsoulos[1] , Dimitris Gavrilis[2], Euripidis Glavas[3]
[1]Department of Computer Science, University of Ioannina, P.O. Box 1186, Ioannina 45110 - GREECE,
[2]Department of Electrical & Computer Engineering, University of Patras, Patras 26500 - GREECE
[3]Department of Communications, Informatic and Management, Technological Educational Institute of Epirus , ARTA - GREECE

***Abstract*— A method which is based on grammatical evolution is presented in this paper for the construction of artificial neural networks (ANNs). The method is capable to construct ANNs with an arbitrary number of hidden levels or even recurrent neural networks. The efficiency of the method is tested on a series of classification and regression problems and the results are compared against traditional neural networks.**

Keywords- Genetic programming, grammatical evolution, neural networks, classification, regression, evolutionary process.

I. INTRODUCTION

Artificial neural networks are well established in the bibliography as approximation tools [1], [2]. They have been used with success for pattern recognition [3], solving differential equations [4] etc. Although, the architecture of the neural network as well as its training algorithm, play the most important role in the approximation and small modifications in either of them can lead to major changes in the efficiency of the network. In most cases, scientists have to experiment with the architecture of the neural network by adding or cutting computation nodes from the network, in order to achieve maximum performance. A survey on pruning algorithms for neural networks is given in [5]. During the past years many methods have been proposed for the construction of neural networks. In [9] a s simulated annealing algorithm was used to optimize neural networks architecture and weights with application to an odour classification problem. Also in [10] a method which is based on Particle Swarm Optimization (PSO) is used for the construction of neural networks. Genetic algorithms has also been applied to the optimization of the architecture of neural networks [11], [12], [13], [14]. This article introduces a method that can produce artificial neural networks using a mapping process governed by a grammar expressed in Backus Naur Form. The proposed method aims to infer the optimal architecture for a given problem accompanied by an optimal

set of weights. Grammatical evolution has been applied successfully to problems such as symbolic regression [6], financial prediction [7] etc. Further details about grammatical evolution can be found in [8]. The rest of this article is organized as follows: in section II the method is described in detail, in section III the efficiency of the proposed method is tested against some data fitting problems as well as classification problems and finally in section IV some conclusions and guidance for further work are presented.

II. DESCRIPTION

The proposed method relies on a series of integer vectors, which will be called chromosomes for the rest of this article. Every chromosome, through a mapping procedure described in [8] that is governed by the BNF grammar of figure 1, is mapped to an artificial neural network with one hidden level and one output. The output of the constructed neural network is a summation of different sigmoidal units and it can be formulated as $N(x,p) = \sum_{i=1}^{H} p_{(d+2)i-(d+1)} \text{sig}(o_i(x) + p_{(d+2)i})$, where $o_i(x) = \sum_{j=1}^{d} p_{(d+2)i-(d+1)+j} x_j$ and $x \in R^d$, $H = \frac{\text{nodes}}{d+2}$ is the number of hidden nodes and the vector p denotes the parameters (weights) of the neural network. The function $\text{sig}(x)$ is the sigmoidal function $\text{sig}(x) = \frac{1}{1+\exp(-x)}$. The constant nodes represents the total number of parameters. The basic steps of the proposed algorithm are the following:

1) Initialization of the population, where every element of each chromosome is initialized randomly in the range [0..255].

Fig. 1. Proposed grammar

```
<S>::=<sigexpr>
<sigexpr>::=<Node>
            |<Node> <sigexpr>
<Node>::=<weight>*sig(<sum>+<bias>)
<sum>::=<weight>*<xxlist>
        |<sum>+<sum>
<xxlist>::=x1 | x2 | ... | xD
<weight>::=<number>
<bias>::=<number>
<number>::=(<digitlist>.<digitlist>)
          |(-<digitlist>.<digitlist>)
<digitlist>::=<digit>
             |<digit><digitlist>
<digit>::=0 | 1 | 2 | 3 | 4
            | 5 |6 |7 |8 |9
```

2) Fitness evaluation. For a given chromosome g the steps for the fitness evaluation are the following:
   a) Apply the mapping procedure of the grammatical evolution, creating a function $f(x)$.
   b) Set $v = 0$.
   c) For every point (x_i, y_i) in the training set $v = v + (f(x_i) - y_i)^2$. The fitness values is set to $-v$.
3) Application of the genetic operations of selection, crossover and mutation. The procedure of selection are performed using tournament selection with tournament size 8. The selection procedure creates a mate pool that will be used in the crossover procedure to create new chromosomes, which will replace the worst chromosomes in the population. In the mutation procedure every element of each chromosome can be changed with a probability $p_m \in$

$[0, 1]$.

4) Terminate either if a maximum number of generations is reached (typical value 2000) or the best chromosome of the population has fitness value below a predefined threshold e (typical value $e = 10^{-7}$). Otherwise jump to step 2.

III. EXPERIMENTAL RESULTS

In order to evaluate the performance of the proposed technique a series of classification and regression problems were used. The proposed technique was issued 30 times for each test problem using different seeds for the random generator each time. Each time the genetic algorithm evolved 2000 generations. In all the experiments the number of chromosomes in the population was set to 500 and the length of the chromosomes to 200. The selection rate was set to 10% and the mutation rate to 5%. The evaluation of the produced chromosomes was made by the FunctionParser programming library [15]. The results from the proposed method are compared against a traditional Multilvel Perceptron trained with the Tolmin optimization method.

A. Classification problems

Wdbc problem : The Winsonsin Diagnostic Breast Cancer dataset (WDBC) contains data from breast tumors. It contains 569 training examples of 30 attributes each that are classified into two categories. In the conducted experiments 303 examples were used for training and 266 for testing. The best constructed neural network was: $f(x) = -40.1\text{sig}(88x_{19} + 19.9x_{29} - 20.1) + 0.99\text{sig}(-0.5x_{23} - 180.2x_{25} - 0.6x_{22} + 98.1)$ with test error 3.01%. The final form has only two nodes and only 5 attributes are used to construct this form. The best traditional neural network had a test error of 5.64% with four hidden nodes.

Ionosphere problem : The Ionosphere dataset contains data from the John Hopkins Ionosphere database. The dataset contains 351 examples of 34 attributes each that belong into two categories. In the conducted experiments 115 examples were used for training and the rest for testing. The best discovered constructed neural network was $f(x) = 0.39\text{sig}(3615.2x_3 - 407.4) + 0.5\text{sig}(588041.3x_5 - 5.8) - 13.8\text{sig}(-6.6x_4 - 9.8) - 3.3\text{sig}(3.7x_4 - 5.6)$ with test error 6.36%. The constructed functional form has only 4 nodes and it uses only 3 from the attributes of the problem. The best traditional neural network had a test error of 12.71% with 3 hidden nodes.

Circular problem : The circular artificial data contains 1000 examples that belong to 2 categories (500 examples each). The data in the first class belong to the inside area of a circle and data of the second class belong to a circular disc outside the first circle. Each example vector has two attributes. It is expanded by adding 3 more attributes generated randomly (noise) using a normal distribution. In the conducted experiments 354 examples were used for training and the rest for testing. The best discovered constructed neural network was $f(x) = 0.8\text{sig}(6.6 - 6.5x_1) + 0.8\text{sig}(6.8x_1 - 20.8) + \text{sig}(7.47 - 7.2x_2) + 1.1\text{sig}(2.9x_2 - 9.0)$ with test error 4.33%. The functional form has 4 nodes and it uses only the two attributes that contains the actual information and not the noise. The best neural network trained by Tolmin method gave an test error of 4.49% with

10 hidden nodes.

B. Regression problems

The Ailerons problem: This data set addresses a control problem, namely flying a F16 aircraft. This problem has 40 attributes and consists of 7150 examples 200 of which were used for training and the rest were used for testing. The best discovered function was $f(x) = 4.9\text{sig}(0.1x_7 - 8.0) - 9.9\text{sig}(9.8x_{37} - 5.0x_9 + 24.5x_{35} + 12.3x_{11} + 0.2x_3 - 9.1)$ with test error 4.4×10^{-8}. The resulting functional form has two nodes and it uses only 6 attributes. The best traditional neural network trained by the Tolmin method had an test error of 4.03×10^{-8} with 3 hidden nodes.

The Pyrimidines problem: This dataset contains 27 attributes and 74 number of patterns. The task consists of Learning Quantitative Structure Activity Relationships (QSARs). The Inhibition of Dihydrofolate Reductase by Pyrimidines. From the above dataset 50 patterns were used for training and 24 for testing. The best discovered function was $f(x) = \text{sig}(3.7x_{26} + 0.4) - 4.5\text{sig}(5.9x_{19} - 37.6x_{11} - 3.2) - 27.5\text{sig}(-9.9x_9 - 4.6) + 5.3\text{sig}(3.7x_{11} - 8.0)$ with test error per point 4.19×10^{-3}. The best neural network optimized by the Tolmin method gave a test error of 6.3×10^{-3} with seven hidden nodes.

IV. CONCLUSIONS

A method which is based on genetic programming for the creation of artificial neural networks was presented in this article. The proposed method can infer not only the architecture of an neural network but it can estimate an optimum set of nodes for a given problem. The method was tested on classification problems as well as data fitting problems. The grammar of the proposed method can be enriched to implement artificial neural networks of higher complexity, such as neural networks with more than two processing levels or even recurrent neural networks. The memory requirements of the proposed method depend on the product of the number of chromosomes and each chromosome's length. The complexity of the method and thus the time required in order to complete depends only on the size of the training dataset and on the population size. The average time for completion for the experiments described above is 4 minutes on an Athlon 2400 with 512 Mb Ram running Debian Linux.

REFERENCES

[1] Hornik K., Stinchcombe M., and White H., Neural Networks 2 (1989) 359.

[2] Cybenko G., Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems 2 (1989) 303-314.

[3] Bishop C., Neural Networks for Pattern recognition, Oxford University Press, 1995.

[4] Lagaris I. E., Likas A., Fotiadis D. I., "Artificial Neural Networks for solving ordinary and partial differential equations", IEEE Trans. on Neural Networks, 9 (1998) 987-1000.

[5] Pruning algorithms - a survey, IEEE Transactions on Neural Networks 4(5): 740-747.

[6] M. O'Neill and C. Ryan, Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, volume 4 of Genetic programming. Kluwer Academic Publishers, 2003.

[7] A. Brabazon and M. O'Neill, "A grammar model for foreign-exchange trading," In H. R. Arabnia et al., editor, Proceedings of the International conference on Artificial Intelligence, volume II, CSREA Press, 23-26 June 2003, pp. 492-498, 2003.

[8] M. O'Neill and C. Ryan, "Grammatical Evolution," IEEE Trans. Evolutionary Computation, Vol. 5, pp. 349-358, 2001.

[9] M.C.P de Souto A.Yamazaki and T.B. Ludernir, Optimization of neural network weights and architecture for odor recognition using simulated annealing. Proceedings of the 2002 International

Joint Conference on Neural Networks, 1:547-552, 2002.

[10] Chunkai Zhang, Huihe Shao, and Yu Li, Particle Swarm Optimization for evolving artificial neural networks, IEEE International Conference on Systems, Man and Cybergenetics, 2000, 4:2487-2490, 2000.

[11] John R. Koza and James P. Rice, Genetic generation of both the weights and architecture for a neural network. International Joint Conference on Neural Networks, IJCNN-91, II:397-404, 8-12, 1991.

[12] Matt Bell, Evolving the structure and weights of recurrent neural network though genetic algorithms. In John R. Koza, editor, Genetic Algorithms and Genetic Programming at Stanford 1999, pages 11–20. Stanford Bookstore, Stanford, California, 94305-3079 USA, 15 March 1999.

[13] Frederic Gruau. Genetic synthesis of modular neural networks. In Stephanie Forrest, editor, Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93, pages 318–325, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.

[14] Joao Carlos Figueira Pujol and Riccardo Poli. Evolution of the topology and the weights of neural networks using genetic programming with a dual representation. Applied Intelligence, 8:73–84, 1998.

[15] Nieminen J. and Yliluoma J., "Function Parser for C++, v2.7", available from http://www.students.tut.fi/warp/FunctionParser

[16] Powel M. J. D., "A Tolerant Algorithm for Linearly Constrained Optimization Calculations", Mathematical Programming 45 (1989), 547