Automatic Recurrent and Feed-Forward ANN Rule and Expression Extraction with Genetic Programming

Julian Dorado, Juan R. Rabuñal, Antonino Santos,
Alejandro Pazos, and Daniel Rivero

Univ. da Coruña, Facultad Informática, Campus Elviña, 15071 A Coruña, Spain
{julian,juanra,nino,ciapazos}@udc.es
danielrc@mail2.udc.es

Abstract. Various rule-extraction techniques using ANN have been used so far, most of them being applied on multi-layer ANN, since they are more easily handled. In many cases, extraction methods focusing on different types of networks and training have been implemented. However, there are virtually no methods that view the extraction of rules from ANN as systems which are independent from their architecture, training and internal distribution of weights, connections and activation functions. This paper proposes a rule-extraction system of ANN regardless of their architecture (multi-layer or recurrent), using Genetic Programming as a rule-exploration technique.

1 Introduction

Artificial Neural Networks (ANN) are systems which are easily implemented and handled. These and other features make them optimal for problem-solving in various areas. However, many developers and researchers avoid their use, since they consider them as "black boxes", that is, they are systems which produce certain response outputs from a series of inputs, while the process through which those outputs are produced remains unknown. For that reason, in fields such as medicine, where their use is highly recommended, people do not consider ANN to be accountable, due to the fact that the reason for their right functioning and the solutions they contribute cannot be explained. An example could be medical diagnosis. A computational system designed for diagnosis should be able to explain the reason for that diagnosis and how it was reached.

Nevertheless, Expert Systems (ES) are able to explain the solution or response achieved, which is their main core and also their guarantee of success. Therefore, this paper tries to develop a system which carries out an automatic extraction of rules from already trained ANN, thus obtaining the knowledge that an ANN obtains from the problem it solves.

Different rule-extraction techniques using ANN have been used so far, always applied to multi-layer ANN due to the fact that they are more easily handled. These networks also have a limited capacity with regard to the knowledge which can be distributed among their connections.

As may be inferred, the extraction of rules and expressions from recurrent ANN is more complicated, due to the fact that past states intervene in neural activation, and

J.J. Merelo Guervós et al. (Eds.): PPSN VII, LNCS 2439, pp. 485–494, 2002.

that their capacity of distributed knowledge is considerably higher than that of multi-layer ANN, since there are no restrictions to neural connectivity. If, besides, recurrent ANN are used in dynamic problems where certain time characteristics such as the prediction of time series intervene, the task of extracting by means of the methods developed so far becomes harder, if not impossible for most of them.
Therefore, the system presented can be applied to every ANN kind. For that reason, the system should comply with a series of requirements [1]:

- ***It should have no ANN architecture requirements***: A rule-extraction mechanism which can work with every type of neural network, including those which are highly interlinked or recurrent.
- ***It should have no ANN training requirements***: Many of the proposed algorithms are based on a given ANN training process for which rule-extraction is designed. Therefore, they are not valid for other training sets.
- ***It should be correct***: It is desirable that the rules describe the ANN as accurately as possible.
- ***It should have a highly expressive level***: Rule language (syntax) characterises the compactness of the extracted symbolic language. Usually, powerful languages are desirable, due to the fact that a very compact and easy to understand rule language can be produced.

2 State of the Art

2.1 Genetic Programming

Some people thinks that Cramer and Fuji, who published on evolving programs in 1985 and 1987 at the very first ICGA conference [2][3], are the pioneers of Genetic Programming (GP). But still others think that Friedberg from 1958 and 1959, who evolved machine language programs [4][5], is really the pioneer.

John Koza created the term which titles the book “Genetic Programming” [6]. This book establishes formally the bases of GP used nowadays. Later, the same author published “Genetic Programming II” [7], and, recently, “Genetic Programming III” [8]. Both explore new possibilities of GP.

Different branches derive form GP. One of the most promising ones with regard to Knowledge Discovery (KD) is that of fuzzy rules [9][10]. This branch derives from the union between fuzzy logic and systems based on rules (SBR). Fuzzy rules can be obtained by means of Evolutionary Computation (EC) with the technique known as Automatically Defined Functions (ADF) [7], which represent an evolution of the concept called “Classical Genetic Programming”.

2.2 ANN Rule Extraction

Several authors have studied the relationship between ANN and fuzzy rules [11] [12] [13]. Most results establish that equivalence by means of a process of consecutive approaches. Apart from being purely theoretical solutions, they require a great number of rules in order to approach the ANN functioning [13]. Jang’s and Sung’s work [11] is rather different, given that they provide an equivalence between radial ANN and fuzzy systems where a finite number of rules or neurons is required, though in this case it is limited to a fixed ANN architecture.

Andrews [14][15] identifies three rule-extraction techniques: “decompositional”, “pedagogical” and “eclectic”. The first one refers to extraction at the neuron level. The second one treats the ANN as a black box, where, by means of applying inputs to the network, a backward to forward analysis of the neurons in the hidden layers is carried out, extracting the corresponding rules. The third one uses the ANN architecture and the input-output pairs as a complement to a symbolic training algorithm.

Towell and Shavlik [16] apply the first technique using the connections between neurons as rules based on simple transformations. This limits extraction to those networks with a given multi-layer architecture and few process elements [14]. Thrun [17] has developed the main approach by means of using the second technique, titled “Validity Interval Analysis” (VIA). The algorithm uses linear programming (SIMPLEX), applying value intervals to each neuron’s activations in each layer. The system extracts “possibly correct” rules through the ANN by means of a backward and forward propagation of those intervals. This algorithm has, in the worst of cases, exponential complexity, due to the fact of using linear programming. Other approaches using the second technique are RULENEG algorithms [18] and DEDEC ones [19], which use an ANN in order to extract rules from another ANN’s training set. However, those rule-extraction techniques which focus exclusively on the training data lose the generalization capacity which ANN have. Other rule-extraction techniques are [20] [21] [1], which are based on previously debated approaches.

GAs have recently been used for finding and extracting ANN, due to the advantages offered by evolutionary techniques for searching in complex systems [22]. The second technique (pedagogical) uses a GA where the chromosomes are multi-condition rules based on intervals or value ranks applied to the ANN inputs. These values are obtained from the training parameters. Wong and Leung have used PG for knowledge discovery from databases (KDD) developing a system called LOGENPRO (Logic grammar based GP) [23]. It uses first order logic to represent knowledge. This is the first aproximation that shows the advantages of GP for KDD.

3 Fundamentals

One of the most important aspects of any rule-extraction system is the optimization of the rules obtained from the ANN analysis. It should be kept in mind that the extracted rules may be contained in general rules, and many of the logical expressions obtained may be simplified if they are written in a different way. Therefore, the optimization of rules consists of simplifying and carrying out symbolic operations on the rules. Depending on the extraction method and on the type of rules obtained, various optimization techniques can be applied. They can be classified into two main groups: imbibed optimization methods and a posteriori methods. The latter are usually a syntactic analysis algorithm applied to the rules obtained in order to simplify them. For instance [24] uses Prolog as programming language for a post-processing of the rules obtained. Imbibed optimization techniques are used for rule-extraction algorithms which intrinsically cause the algorithm to produce rules which are more and more optimal. An example may be the technique of depth penalization used in GP. Conceptually, when the adaptation level of a GP individual is evaluated (tree) its capacity is reduced a certain degree according to the number of terminal and non-

terminal nodes that the tree has. Thus the existence of simple individuals is dynamically fostered. Therefore, if we are searching for rules (syntactic trees), the appearance of simple (optimization) rules is intrinsically favoured.

Another thing to be taken into account when applying the extraction algorithm is its modus operandi. As previously discussed, extraction techniques can be classified into three main groups: "decompositional", "pedagogical" and "eclectic". EC has been applied in this paper, and specifically GP as building algorithm of a syntactic tree which reflects a set of rules as similar as possible to the functioning of an ANN. A symbolic regression has been applied to the input-output patterns. These patterns are input sequences applied to an ANN and the outputs obtained from it. This type of technique can be termed as "pedagogical", where the ANN is treated as a "black box". This is an advantage, given that it is not necessary to know how an ANN works internally. However, a rule-extraction algorithm which can work with "black box" structures should be able to implement some kind of mechanism which allows a priori incorporation of the knowledge obtained from the "black box", thus reducing considerably the search space of the system rules. These structures are known as "grey box". This is possible thanks to the fact that in GP the number and type of terminal and non-terminal elements which intervene in the search process can be determined. For instance, if we know that an ANN carries out classification tasks, the type of terminal nodes can be determined as Boolean, avoiding floating point levels. This offers a great advantage, given that all the possible combinations of mathematical operations can be eliminated beforehand.

4 Description of the System

The main method proposed for the extraction of ANN rules is the use of EC, due to the fact that it has been proved to be very useful at search tasks, where the solution space increases exponentially with regard to the problem to be solved. The use of GP is proposed since it offers the advantage of having a way of representing and structuring information by means of a semantic tree. This tree diagram is a natural way of representing a rule which can be easily understood by human beings.

The proposed system will be tested contrasting its correct functioning with other existing extraction techniques based on classification tasks. The cases are the breast-cancer diagnosis and the lethal hepatitis one. In these cases, the direct extraction of rules from those data will be proven with the purpose of extracting the rules pertaining to each network by means of training ANN. Finally, the algorithm will be tested with recurrent ANN for time-series prediction tasks. A RANN trained for predicting a laboratory chaotic time series (such as the Mackey-Glass) will be used.

In each case, the work is initially based on obtaining an ANN for solving the problem. Once the ANN are designed and trained, the same test and training values are used for generating a second data pattern which will be used for finding the rules acquired by the ANN in the training process (Fig.1). The rule-extraction algorithm, as discussed before, is based on GP. This search technique allows problem solving by means of the automatic generation of algorithms and expressions. These expressions are codified in the shape of a tree. In order to create this tree, we must specify which nodes will be terminals (leaves) and which will be functions (non terminals). The difference is that some of them will be able to have offspring and the others will not.

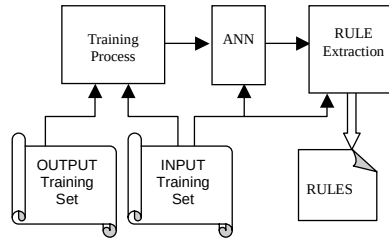

Fig. 1. Rule extraction process

When terminal and functions operators are specified, it is necessary to specify their types: each node will have a type, and the functions will require a specific type to their offspring [25]. This ensures that the trees thus generated satisfy the user's grammar. Besides, both specified operator sets must comply with two requirements: closure and sufficiency, i.e. it must be possible to build correct trees with the specified operators, and the solution to the problem must be able to be expressed by means of those operators. Depending on the problem to be solved, mathematical and trigonometrical functions (sine, cosine, square,…), logic operators (AND, OR, NOT) have been used as operators, together with the typical operations +, -, *, and % which is the protected division (avoid dividing by zero). Thus, logic operators have been basically selected for classification tasks, dispending with mathematical operations. On the opposite side, it is necessary to use them for prediction, due to the nature of the outputs of the ANN which deal with this type of problem.

5 Results

5.1 Classification Problems

ANN have shown their extreme usefulness for tasks in which having an input sequence we must decide whether those values correspond to a certain classification. As it was debated before, various problems of medical diagnosis have been used in order to train ANN so that they produce it. The first data set used was the detection of lethal hepatitis cases. For this purpose we used a database obtained from UCI [26]. It contains 155 examples for two classifications: 32 deaths, and 123 alive cases. There are 19 attributes, 13 of which are Boolean while 6 have discrete values. An multi-layer architecture ANN with 9 neurons in only one hidden layer and with tangent hyperbolic activation functions was trained with these examples, obtaining a fitness of 100% of cases. The extraction of rules has been applied to this ANN and the fitness value obtained is the correct classification of 98.75% of cases.

(IF X_2 THEN 0.8826
ELSE (IF ((IF X_2 THEN 0.8859
ELSE X_{18})<0.6247)
THEN (IF (IF ((IF X_6 THEN 0.9800
ELSE 0.6205)>0.9380)
THEN (NOT X_{12})
ELSE X_7)
THEN X_{18}
ELSE 0.8826)
ELSE (IF X_{13} THEN X_{18}
ELSE 0.2338))) (1)

(X_1) age
(X_2) sex: *men*,*woman*
(X_3) steroid: *no*,*yes*
(X_4) antivirals:*no, yes*
(X_5) fatigue: *no, yes*
(X_6) malaise: *no, yes*
(X_7) anorexia: *no, yes*
(X_8) liver big: *no, yes*
(X_9) liver firm: *no, yes*
(X_{10}) spleen palpable: *no, yes*
(X_{11}) spiders: *no, yes*
(X_{12}) ascites: *no, yes*
(X_{13}) varices: *no, yes*
(X_{14}) bilirubin
(X_{15}) alk phosphate
(X_{16}) sgot
(X_{17}) albumin
(X_{18}) protime
(X_{19}) histology: *no, yes*

This results were obtained with the following operators:

- *Constants*: 20 random values in the range [0,1]
- *Variables*: 19 inputs: 13 boolean and 9 continuous
- *Operators*: <, >, =, AND, OR, NOT, IF-ELSE on Boolean and Real values

Table 1. Comparison to other existing rule extraction methods

Method	Accuracy	Ref.	Method	Accuracy	Ref
OUR	98.75 %		ASR	85.0 %	[27]
C-MLP2LN	96.1 %	[24]	FDA	84.5 %	[27]
k-NN, k=18, Manhattan	90.2 %	[24]	LVQ	83.2 %	[27]
FSM + rotations	89.7 %	[24]	CART	82.7 %	[27]
LDA	86.4 %	[27]	MLP with BP	82.1 %	[27]
Naive Bayes	86.3 %	[27]	ASI	82.0 %	[27]
IncNet + rotations	86.0 %	[28]	LFC	81.9 %	[27]
1-NN	85.3 %	[27]	Default	79.4 %	

The next data set corresponds to the detection of breast cancer. We also have a database obtained from UCI [26]. It contains 699 examples for two classifications: 458 cases of benign cancer (65.5%) and 241 cases of malign cancer (34.5%). There are 9 attributes, all of then are discrete.

Various ANN have been trained. However, 100% of classifications was never reached. The rate of success was 98.28% with one hidden layer with 7 neurons and neurons with linear activation function. The same architecture with tangent hyperbolic activation functions has improved the success rate up to 98.68%.

The value of the best fitness obtained for this latter ANN is a right classification in 99.71% of cases (using the outputs produced by the ANN). However, in order to draw a right comparison to other rule extraction techniques, the algorithm is directly applied to the set of initial input-output patterns (dispensing with the ANN outputs). In this case, the global fitness value obtained is the correct classification of 99.28% of cases. The rules obtained are the following (2):

```
(IF (((0.9<X1) OR  (IF (X1>0.4)
                    THEN (((0.3>X7) AND (X1<>X4)) AND (X6>0.2))
                    ELSE FALSE))
                OR (((0.3<X2) AND (X5>0.4))
               AND  (IF (X9<>X4)
                    THEN (X1<>X4)
                    ELSE (X3>0.4))))
THEN   (IF ((X3>0.4) OR (X5>0.4))
        THEN (0.0<>X6)
        ELSE ((0.9<X1) OR (0.4<X6)))
ELSE   (IF  (IF (0.4>X8)
            THEN (0.3>X3)
            ELSE (X3=0.6))
        THEN (0.9<X1)
        ELSE   (IF (X3>0.4)
                THEN  (IF (0.3<>X4)
                      THEN (0.4>X8)
                      ELSE (0.4<X6))
                ELSE (X3>X5))))
```

(2)

(X1) clump thickness
(X2) uniformity of cell size
(X3) uniformity of cell shape
(X4) marginal adhesion
(X5) single epithelial cell size
(X6) bare nuclei
(X7) bland chromatin
(X8) normal nucleoli
(X9) mitoses

Table 2. Comparison to other existing rule extraction methods

Method	Accuracy	Ref.	Method	Accuracy	Ref
OUR	99.28 %		Bayes (pairwise dependent)	96.6	[27]
C-MLP2LN	99.0 %	[24]	Naive Bayes	96.4	[27]
IncNet	97.1	[28]	DB-CART	96.2	[29]
k-NN	97.0	[24]	LDA	96.0	[27]
Fisher LDA	96.8	[27]	LFC, ASI, ASR	94.4-95.6	[27]
MLP with BP	96.7	[27]	CART	93.5	[29]

5.2 Forecast of Time Series

It is necessary to use RANN architectures for the prediction of time series and for modelling this type of problems. The extraction of rules from ANN with recurrent architecture has an additional challenge, since these ANN are characterised by their huge capacity of representation and distributed knowledge among their connections. This can be specifically applied to time and dynamic problems. The problem to be solved will be the prediction of a classical chaotic laboratory time series: the Mackey-Glass series [30]. The following results show that the rules to be obtained from this ANN should incorporate mechanisms for treating time values. Therefore, non-terminal nodes representing mathematical and trigonometrical operations will be used, together with input variables at previous n moments (X_n). Most of these time series cases are structures with a single input and a single output. The input corresponds to a number value at the t moment, while the system's output is the prediction of the number value at t+1. The Mackey-Glass equation is an ordinary differential delay equation (3).

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t) \tag{3}$$

Choosing τ = 30, the equation becomes chaotic, and only short-term predictions are feasible. Integrating the equation (3) in the rank [t, t + δt] we obtain:

$$x(t+\Delta t) = \frac{2 - b\Delta t}{2 + b\Delta t}x(t) + \frac{\alpha\Delta t}{2+b\Delta t}\left[\frac{x(t+\Delta t-\tau)}{1+x^c(t+\Delta t-\tau)} + \frac{x(t-\tau)}{1+x^c(t-\tau)}\right] \tag{4}$$

The first step is obtaining a RANN which emulates the behaviour of the time series. The RANN that we used has three neurons with tangent hyperbolic activation function with total interconnection. The training files used correspond to the first 200 values of the time series (Fig.3). The RANN resulting from the training process which has yielded the least mean square error (MSE=0.000072) may be seen in Fig.2.

Once we have obtained the RANN, we try to obtain by means of symbolic regression the rules and the expressions which direct its functioning. For this purpose we have used a test file containing the first 1000 values of the time series. These 1000 values are transferred to the RANN, obtaining the corresponding outputs. Using the input-output file, we run the GP algorithm.

Different combinations of terminal and function elements and GP parameters have been tried, and the following have been used:

- *Arithmetic functions*: +, -, *, % (protected division)
- *Constants*: 10 random values in [0,1]
- *Variables*: X_n, X_{n-1}, X_{n-2}, X_{n-3}
- *Selection algorithm*: Tournament
- *Population size*: 1000 individuals
- *Crossover rate*: 95%
- *Mutation rate*: 4%
- *Parsimony level*: 0.0001

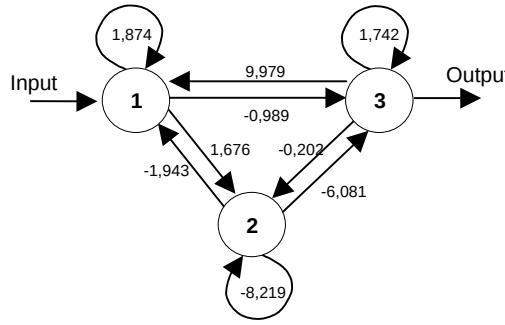

Fig. 2. RANN that emulate the Mackey-Glass function

Besides, the elitist strategy has been used, not allowing the loss of the best individual. The rule expressed as a mathematical function is the following:

$$((X_n * ((((((X_n * (X_n * X_{n-2})) \% X_{n-2}) * (((0.9834 * (((((X_n * (X_{n-2} \% X_{n-3})) * X_{n-3}) * X_{n-3}) \% X_{n-3}) \% (X_n \% X_{n-3}))) \% X_{n-3}) \% ((X_{n-2} * X_{n-2}) \% X_{n-3}))) \% X_{n-3}) * X_{n-2}) \% X_{n-2})) \% ((X_n * X_{n-2}) \% X_{n-3}) * 0.9834)) \quad (4)$$

This function obtains a fitness value (normalised) on the 1000 values produced by the RANN of 0.0029. The next graph compares the values produced by the function forecast (4) and the RANN forecast.

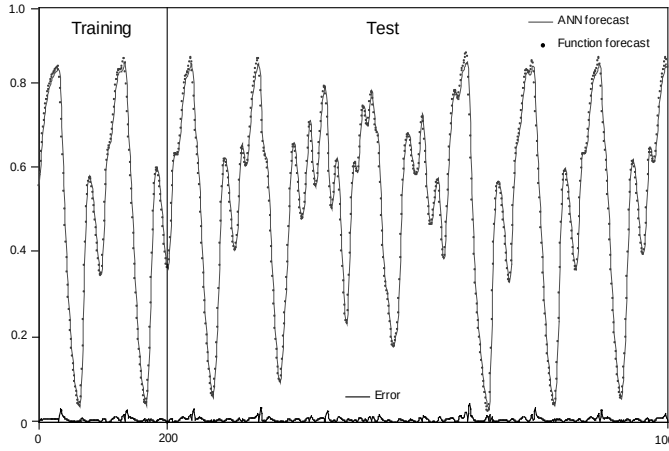

Fig. 3. Comparative between the RANN forecast and the function forecast (4)

6 Conclusions

As inferred from the results presented, the GP-based rule extraction algorithm considerably improves the existing methods, being applicable to any ANN architecture, whether recurrent or feed-forward. Depending on the task for which the ANN has been designed, the types of operators on which GP works might be varied.

As discussed in the introduction, the four features required by a rule-extraction system are faithfully approached in this case. By means of applying EC, ANNs are treated as black boxes, where only the inputs and the outputs they produce are considered. And it does not depend neither on their architecture nor on their training algorithm. The rules extracted from the ANN are so similar to the ANN's functioning as the fitness value produced by the algorithm, and as expressive as the semantic level used by the GP's codified expression trees. Therefore, we may say that GP-based algorithms fit the needs of the rule-extraction systems.

7 Future Works

A possible development is to apply GP not only to obtain rules, but also operability ranks of ANN. It would be like a validation system for the functioning of an ANN for the problem to be solved. Not only the initial training set which carries out the adjustment should be used for this purpose, but also new input data sets should be created simulating all the possible variations of values in which could enter the ANN.

Another future development will be the analysis of the different parameters intervening in the algorithm's correct functioning, according to the type of problem solved by the ANN. It should be treated not as a black box, but as a grey one, where, for instance, the ANN's activation function is known, being incorporated as one of the operators of GP, and analysing which are the rules extracted by this operator.

In order to accelerate the rule-extraction process, it is possible to use a network of computers and so the rule search will be distributed and concurring, exchanging rules.

Acknowledgements

This work has been supported by the Univ. A Coruña project "Extracción de reglas de RNA mediante CE" and CICYT TIC2000-0120-P4-03 of the Spanish government.

References

1. Tickle, A.B.; Andrews, R.; Golea, M.; Diederich, J. "The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks". IEEE Transaction on Neural Networks, vol. 9 nº 6, pp 1057-1068, 1998.
2. Cramer, N.L. "A Representation for the Adaptive Generation of Simple Sequential Programs", Grefenstette: Proc. of 1st International Conference on GA, 1985.
3. Fujiki C. "Using the Genetic Algorithm to Generate Lisp Source Code to Solve the Prisoner's Dilemma", International Conf on GAs, pp. 236-240, 1987.
4. Friedberg R. "A learning machine: Part I", IBM Journal, pp. 2-13, 1958.
5. Friedberg R.M., Dunham B., North J.H. "A learning machine: Part II", IBM Journal of Research and Development, 3(3) 282-287, 1959.

6. Koza J. “Genetic Programming. On the Programming of Computers by means of Natural Selection”. The Mit Press, Cambridge, Massachusetts, 1992.
7. Koza J. “Genetic Programming II: Automatic Discovery of Reusable Programs”. The Mit Press, Cambridge, Massachusetts, 1994.
8. Koza J., Forrest H., Andre D., Keane M. “Genetic Programming III: Darwinian Invention and Problem Solving”. Morgan Kaufmann Publishers, San Francisco, 1999.
9. Fayyad U., Piatetsky-Shapiro G., Smyth P., Uthurusamy R.: “Advances in Knowledge Discovery and Data Mining”. AAAI/MIT Press, 1996
10. Bonarini A.: "Evolutionary Learning of Fuzzy Rules: Competition and Cooperation", Fuzzy Modelling: Paradigms and Practice, W. Pedrycz (Ed.), Kluwer Academic Press, 1996
11. Jang J., Sun C. “Functional equivalence between radial basis function networks and fuzzy inference systems”. IEEE Transactions on Neural Networks, vol. 4, pp 156-158, 1992.
12. Buckley J.J., Hayashi Y., Czogala E. “On the equivalence of neural nets and fuzzy expert systems”, Fuzzy Sets Systems, Nº 53, pp 129-134, 1993.
13. Benítez J. M., Castro J. L., Requena I. “ Are artificial neural networks black boxes? “ IEEE Transactions on Neural Networks, vol. 8, nº 5, pp 1156-1164, 1997.
14. Andrews R. Diederich J., Tickle A. “A Survey and Critique of Techniques For Extracting Rules From Trained ANN”. Knowledge Based Systems 8, pp 373-389, 1995.
15. Andrews R., Cable R., Diederich J., et al: “An evaluation and comparison of techniques for extracting and refining rules from ANN”, QUT NRC Technical report, 1996.
16. Towell G., Shavlik J. W. “Knowledge-Based ANN”. AI, 70, pp 119-165, 1994.
17. Thrun S. “Extracting rules from networks with distributed representations”. NIPS, G. Tesauro, D. Touretzky, T. Leen (eds), MIT Press, 1995.
18. Pop E., Hayward R., Diederich J. “RULENEG: Extracting Rules from a Trained ANN by Stepwise Negation”. Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report, 1994.
19. Tickle A. B., Orlowski M., Diedrich J. “DEDEC: A methodology for extracting rules from trained artificial neural networks”. Queensland Univ. of Technology, Neurocomputing Research Centre. Technical report, 1996.
20. Chalup S., Hayward R., Diedrich J. “Rule extraction from artificial neural networks trained on elementary number classification task”. Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report, 1998.
21. Visser U., Tickle A., Hayward R., Andrews R. “Rule-Extraction from trained neural networks: Different techniques for the determination of herbicides for the plant protection advisory system PRO_PLANT”. Proc. of the rule extraction from trained ANN workshop, Brighton, UK, pp 133-139. 1996.
22. Keedwell E., Narayanan A., Savic D. “Creating rules from trained neural networks using genetic algorithms”. IJCSS, vol. 1, Nº 1, pp 30-42. 2000.
23. Wong M.L., Leung K.S.: “Data Mining using Grammar Based Genetic Programming and Applications”, Kluwer Academic Publishers, 2000.
24. Duch W., Adamczak R., Grabczewski K.: “A new methodology of extraction, optimisation and application of crisp and fuzzy logical rules”, IEEE Trans. on N.N., vol. 11, nº 2, 2000.
25. Montana D.J.: “Strongly Typed Genetic Programming”, Evolutionary Computation, The MIT Press, pp. 199-200, Cambridge, MA, 1995.
26. Mertz C., Murphy P.: UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-data-bases.
27. Ster B., Dobnikar A.: “Neural networks in medical diagnosis: Compararison with other methods”, A. Bulsari et al. eds, Proc. Int. Conf. EANN’96, pp. 427-430, 1996.
28. Jankowski N., Kadirkamanathan V.: “Statistical Control of RBF-like Networks for Classification”, 7th International Conf. on ANN, pp. 385-390, Lausanne, Switzerland, 1997.
29. Shang N., Breiman L.: “Distribution based trees are more accurate”, Int. Conf. On Neural Information Processing, vol. 1, pp. 133-138, Hong Kong, 1996.
30. Mackey M., Glass L.: “Oscillation and chaos in physiological control systems”, Science, pp. 197-287, 1977.