Hierarchical Evolutionary Algorithm in the Rule Extraction from Neural Network

Urszula Markowska-Kaczmar, Roman Zagórski

Wroclaw University of Technology, Wyb.Wyspianskiego 27, 50-270 Wroclaw, Poland

Kaczmar@ci.pwr.wroc.pl

Abstract

This paper describes a method of extracting rules from trained neural network based on two-level hierarchical evolutionary algorithm. Evolutionary algorithm on the lower level extracts one rule at a time. Those rules are the base for higher level evolutionary algorithm to search for whole set of rules. The proposed method has been tested on five public domain data sets and the results have been compared with other rule extraction methods.

1. Introduction

Neural networks are very attractive technique because of their ability of generalization. However, they include their 'knowledge' about solved problem in the form of numerical weights and interconnections, which is not comprehensible for the user. In several application domains (for example in medical problems) the explanation of the neural network (NN) solution is strongly desirable to ensure more trust in the response given by the network. This explanation can be seen as a method of the neural network validation or as an automatic knowledge acquisition. These are the reasons of the growing interest in extracting knowledge from neural network.

The number of different approaches to the rule extraction from the neural network is proposed. They can be divided into three groups. Local methods try to analyse an activity of a single neuron in the hidden and output layers. They describe it in a form of rules. Then rules are concatenated so as a final result rules showing a dependence between output and inputs of NN are obtained. In these methods parameters of NN are very intensive exploited. We can list FullRe [11], Subset, M of N [13] and others [7, 8, 9] among these methods. Different approach is used in global methods. They concentrate on behaviour of a whole network treating it as a black box. We can name VIA [12] and other algorithms [4] among this category. Third possibility is called eclectic method and it combines some features from both of previous mentioned methods. A detailed survey of existing methods of knowledge extraction from the neural network can be found in [1, 4]. All of the methods have their advantages and drawbacks. For example some of them are dedicated to the problems with binary or nominal attributes. Others need special training procedure or retraining to extract rules. So the motivation to search an efficient method of the rule extraction from trained neural network, which would be independent on the type of attributes, the NN architecture or training procedure still exists. This paper presents our method of the rule extraction from neural network called GARulExNN (Genetic Algorithm Rule Extraction from Neural Network)[1].

2. The Proposed Method of the Rule Extraction from Neural Network

We have developed the method of a rule extraction from trained neural network based on hierarchical evolutionary algorithm inspired by paper [6]. We have focused on the rule extraction from neural networks because NNs have good tolerance to noise. In our approach neural network is treated as an oracle producing patterns to evaluate rules searched by the evolutionary algorithm. In the above presented taxonomy the proposed method belongs to the global ones. The essence of our approach is composed of two levels of evolutionary algorithms called EA_1 (lower) and EA_2 (upper). The EA_1 is responsible for searching the single rules describing a performance of neural network, while the role of the upper evolutionary algorithm is to search a set of rules, which with the best fidelity describes the performance of the neural network. The motivation behind the use of Two Eas was so that EA_1 produces

[1] This work was supported by Polish Committee for Scientific Research under grant number 4 T11 E 02323

rules with the good accuracy, while EA_2 minimizes the number of rules, what increases the comprehensibility of rules. The similar result could be also obtained by a single EA but by encoding a set of rules in one chromosome. The class of problems is limited to the classification.

2.1. EA_1 – The Lower Level Evolutionary Algorithm

The lower evolutionary algorithm concentrates on searching single rules, that are the base for EA_2 to search for complete set of rules. In EA_1 Michigan

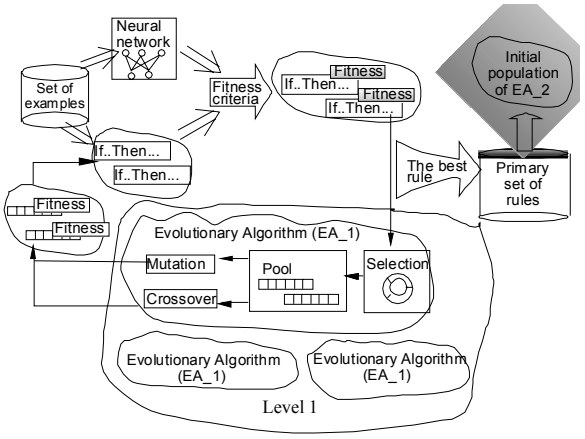

Fig. 1. Schema of the rule extraction by lower level evolutionary algorithm

approach is implemented, what means that in one individual one rule is coded. We assumed that the lower level EA will consist of parallel, independently evolving populations, where each of them is searching rules from one class (Fig. 1.). To implement evolutionary algorithm first of all the form of individual, genetic operators and fitness function have to be design. The details of the essential part of EA_1 will be presented below in this section

Each chromosome at the EA_1 level represents one rule that is given by Exp. 1.

$$\text{If } p_1 \text{AND } p_2 \text{ AND.... THEN } k, \qquad (1)$$

where each p_i denotes i-th premise of the rule corresponding to the given neural network input. For discrete and continuous attributes premise describes the condition that the value of the attribute belongs to the interval $(v_1;v_2)$ according to Exp. 2.

$$attr_i \; IN(v_1;v_2) \qquad (2)$$

For a nominal or Boolean attribute premise defines a value, which the attribute takes (exp 3.).

$$attr_i = value_k \qquad (3)$$

Each chromosome is composed of the genes coding premises and a single gene coding the conclusion (Fig. 2). As a result of a binary flag staying before the code of premise the rules can take a different length. That means we have the ability to obtain rules with variable number

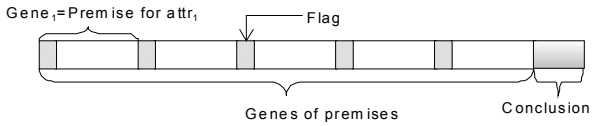

Fig.2. The form of chromosome in the EA_1

of premises. The premise is included in the body of the rule if its flag is set to 1, only. The position of gene coding a premise in the chromosome corresponds to the neural network input. For each type of attribute special kind of gene is designed (Fig.3.). Gene for a binary attribute consists of two elements – flag (A) and the value (Value) of an input attribute (Fig. 3a), which can take one of two values: *false* or *true*. For nominal attributes the form of chromosome is similar, but the value (Value) can take one of the allowed values.

Figure 3b presents gene of premise of the real type of attribute (continuous or discrete). It is composed of flag (A) and limits of a range (X_1, X_2), to which belongs the value of the attribute.

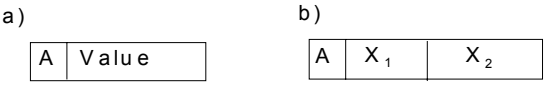

Fig. 3. Types of gene in GARulExNN

Evolutionary algorithm EA_1 uses the standard genetic operators. The mutation operator applied to the flag changes its value to the opposite. In the case when the mutation is performed for gene containing premise describing discrete or continuous attribute the value setting the boundary of interval is substituted by other random value that belongs to the range of allowed values. The only difference between mutation for continuous and discrete attribute is that for continuous attribute it chooses one random value from the allowed range while for discrete attributes the new value (after mutation) is one of the discrete values from the range allowed for this attribute. Mutation is not permitted for gene containing the conclusion. The crossover operator is one-pointed. It is performed by exchanging genes between two individuals from the same class.

The fitness function in GARulExNN promotes the individuals, which cover the most examples from correct class and less examples from improper class. In this context correct classification means that the rules classify the examples in the same way as neural network does. In other words we maximize the extent of covering of examples in a given class and minimize the extent of inconsistent covering of training examples. The fitness

function makes allowance for the length of the rule (a number of premises), as well. The formal form of the fitness function for EA_1 is presented by Eq.4.

$$Fitness_{EA\_1} = \frac{\alpha \frac{TP}{TP + FN} + \beta \frac{C_{\max} - C}{C_{max}}}{k} \quad (4)$$

In the equations (4), (5), (6), (7), (8) *TP* is the number of examples covered by the rule that have the class defined by neural network the same as predicted by the rule, *FP* is the number of examples covered by the rule where class predicted by the rule is different from the one assigned by NN, *FN* is the number of examples that are not covered by the rule but they have class labelled by neural network as predicted by the rule, *TN* is the number of examples that are not covered by the rule and their class determined by neural network is indeed different from one predicted by the rule, C_{max} is the maximum number of premises in the rule (the number of the input attributes) whereas *k* is a penalty for improper classification and is described by Exp. 5.

$$k = \begin{cases} \gamma\, FP \text{ if } \gamma\, FP \geq 1 \\ 1 \text{ in the opposite case} \end{cases} \quad (5)$$

In expression 5. *k* can not be less then 1 as it would increase the value of the fitness function. In the above fitness function both criteria: accuracy and comprehensibility are expressed.

To create new generation in GARulExNN roulette wheel is applied. In this case the probability of the choice to the crossover is proportional to the fitness value. As it was mentioned, individuals in the lower level create the subpopulations. For each class one subpopulation is formed. EA in this population is searching for rules describing examples from this class. Populations are evolving independently through the assumed number of generations, which is set by the user. At the end, from each population the best individual is chosen and after decoding its rule he is put in the set of rules, which we call primary set of rules (Fig 1.) and examples covered by that individual are marked as 'covered'. Then the process of searching single rules for uncovered examples is repeated with the new random population for each class. It is performed as long as there is any example not yet covered.

2.2. EA_2 – The Upper Level Evolutionary Algorithm

The upper level evolutionary algorithm is responsible for searching optimal set of rules. It works with initial population formed on the base of rules delivered by EA_1. It means that EA_2 instead of random initial population starts with the population created on the base of rules obtained from EA_1.In this case one individual represents a set of rules. In order to obtain a variable number of rules the flag before each rule is implemented, as well (Fig.5.).At this level one-pointed crossover operator is implemented, as well. The intersection point can not lie inside the rule. Mutation operator at this level is similar to the lower level, but it introduces two new features. First thing is that a flag for the rule can be set or

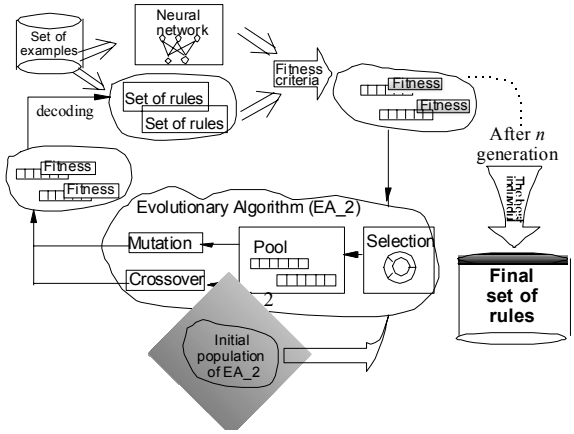

Fig. 4. Schema of the extraction of the final set of rules by EA_2

unset, which means a whole rule can be included or excluded from the decoded set of rules. Second modification applies to the values of premises in the rule. For continuous and discrete attributes change of value can not exceed given constraint, which is computed as percentage of the length of the adequate domain (e.g. 10%). This was made to prevent damages to the rule that is usually already quite good and choosing new random value would rather decrease its performance. This method of changing values can not be applied to the nominal or Boolean domains as their values usually do not have order. Fitness function at this level maximises the predictive accuracy and brings the penalty for false classification. It gives the possibility of affecting the number of rules in the final set of rules. The fitness function is given by Eq. 6.

$$Fitness_{EA\_2} = \frac{\alpha \frac{\sum_{i=1}^{n} TP_i}{\sum_{i-1}^{n} (TP + FN)} + \beta \frac{R_{\max} - R}{R_{max}}}{k}, \quad (6)$$

where R_{max} – is the maximum number of rules. It is equal to the number of rules produced by EA_1, *R* is the number of active rules in the individual, *n* is the number of classes, the meaning of the remaining symbols is the same as in Eq. 4 and 5.

The creation of the initial population in EA_2 consists of encoding in the chromosome all rules acquired by EA_1

and setting flags in random way. Then a selection by

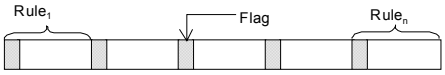

Fig.5. The form of chromosome in the EA_2

roulette wheel takes place. In every generation individuals containing set of rules are evaluated. EA_2 works for a given number of generations and then the best individual decodes its rules, which is the final solution. This set of rules is tested how good they are in generalization (fig. 4.).

3. The performance of GARulExNN

The algorithm of GARulExNN's performance is presented in table 1.

It starts from phase, where EA_1 is active. As it was mentioned EA_1 is composed of parallel evolutionary algorithms searching for rules describing given class. One chromosome here represents one rule. At this level we have as many evolutionary algorithms as many classes exists in the considered problem. In each generation rules are evaluated on the base of the examples produced by the neural network. Each evolutionary algorithm works for assumed number of generations. Next the best individual from each evolutionary algorithm is decoded and passed to the primary set of rules. All covered examples are marked and process is repeated until uncovered examples still exist. If this condition is fulfilled phase EA_2 begins. EA_2 focuses on optimisation of the primary set of rules found by EA_1. Chromosome ofEA_2 contains the set of rules that in the best way describes the performance of trained neural network. Initial population is formed on the base of primary set of rules. The maximum number of genes containing rules in the chromosome is equal to the number of rules in the primary set. However random value of flag staying before rule gives the individuals, which differ from the primary set of rules. During an evolution by applying genetic operators new generation of individuals are formed. In each generation individuals are evaluated in the similar way as in the phase EA_1.

4. Experimental Study

In our experiments we used data sets obtained from *UCI Machine Learning Repository* [3]. These were *Iris, Wine, Monks-1, Monks-3* and *Breast Cancer Wisconsin*. Table 2 shows detailed information about these data sets. The last column of table 2 shows information about classification of all examples after

Table 1. The algorithm of the GARulExNN

Phase EA_1
Create random Initial Populations
  For each EA_1_i do
    While there exists an example in *i*-th class, which is not marked as covered by any rule do
      For a given number of generations do
        Evaluate individuals
        Apply genetic operators
        Create new generation
      End {for}
    Bring the best individual to the primary set of rule
    Mark examples covered by the best individual
      End {while}
End {for}
Phase EA_2
Unmark covered examples
Create an initial population on the base of primary set of rules
For a given number of generations do
      Evaluate individuals
      Apply genetic operators
      Create new generation
End {for}
The best individual denotes the final set of rules
End

the training of neural network was completed. Our task was to prove how good the set of rules can describe the behaviour of trained neural network. This was measured using *predictive accuracy* of the extracted set of rules defined by Eq. 7 and *fidelity* defined by equation 8. In

$$accuracy = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i-1}^{n} (TP + FN)}, \qquad (7)$$

$$fidelity = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i-1}^{n} (TP + FP)}, \qquad (8)$$

both of those equations *n* stands for the number of classes in a data set. *Fidelity* in Eq. 8 is expressed by relation of the number of instances correctly classified by the set of rules to the number of instances classified.

Any example can be classified by the neural network but this statement is not true for a set of rules where an example can remain unclassified. This unclassified example causes decreasing of *accuracy* but has no influence on *fidelity*. In the context of the rule extraction from the neural networks *'correct classification'* means that the set of rules predicts the same class for a given instance as the network does.

Table 2. Experimental data sets from UCI Repository [3]

Dataset	***Description of an instance***	***Class***	***Number of instances***	***Number of instances after the neural network training***
Iris (150 instances)	*4 continuous attributes*	*Setosa*	50	50
		Versicolour	50	46
		Virginica	50	54
Wine (178 instances)	*13 continuous attributes*	*1*	59	59
		2	71	71
		3	48	48
Monks-1 (432 instances)	*Attributes treated as discrete values: 3 attr. with values {1,2,3} 2 attr. with values {1,2} 1 attr. with values {1,2,3,4}*	*0*	216	216
		1	216	216
Monks-3 (432 instances)		*0*	204	204
		1	228	228
Breast cancer Wisconsin (683 instances)	*9 attributes treated as discrete values {1,2,…,10}*	*Benign*	444	437
		Malignant	239	246

In our tests we achieved the best results with probability of crossover around 40% (both levels), probability of mutation around 5-10% (both levels), fitness function parameters (Eq. 4, 5, 6) *alfa=2, beta=0,05-0,1, gamma=3-4* (both levels). The number of generations and the size of population was various but for the *Breast Cancer Wisconsin* data set those parameters were greater than in the rest of the data sets. All of the tests were made using *10-fold cross* validation technique. The results of these experiments for all data sets are summarized in table 3, where *BCW* stands for *Breast Cancer Wisconsin*. *Predictive accuracy* is a measure of *accuracy* for the testing examples whereas *accuracy* was computed using all data set (training and testing examples). The same distinction may be applied to the *fidelity*. Table 4 presents results of GARulExNN for all data sets in the context of *fidelity*. Last column of tables 3 and 4 shows statistics for 'the best found set of rules'. As it was mentioned we used *10-fold cross* validation technique so we extracted 10 final sets of rules. Among those 10 sets we chose one that had the highest *accuracy* (not *predictive accuracy*) and defined that set as 'the best'. Usually there were couple of the rule sets with the highest *accuracy*.

Table 5 shows the change of the example set of rules found by EA_1, which occurred after execution of EA_2. Example was taken from 'the best found set of rules' for *Iris* data set and the final set of rules was obtained as 'the best during evolution'. Rules in the part EA_2 of table 5 are numbered respectively to the ones in the EA_1 part. It can be easily noticed in table 5 that EA_2 removed first two rules and made some changes to the values of premises in rules 3 and 5. It illustrates the ability of EA_2 to optimise the set of rules found by EA_1. Usually EA_2 creates sets of rules with less number of rules. The value of *accuracy* for the final rule sets is less than *accuracy* for the rules obtained by EA_1 but the value of *fidelity* is rather stable. This leads to conclusion that final sets of rules would either correct classify an

Table 3. The results of GARulExNN in the term of *accuracy* rate

Dataset	***Neural network***		***Primary set of rules (EA_1)***		***Rules obtained 'after the last generation' (EA_2)***		***Rules obtained as 'the best individual during evolution' (EA_2)***		***The best found set of rules (EA_2)***	
	Predictive accuracy	*Training accuracy*	*Predictive accuracy*	*Number of rules*	*Predictive accuracy*	*Number of rules*	*Predictive accuracy*	*Number of rules*	*Predictive accuracy*	*Training accuracy*
Iris	0,933	0,978	0,960	8,9 ± 0,407	0,920	5,8 ± 0,389	0,960	7,5 ± 0,453	1	1
Wine	1	1	0,904	14 ± 0,537	0,803	9,8 ± 0,389	0,882	11,3 ± 0,517	0,889	0,994
Monks-1	1	1	1	8,3 ± 0,153	0,988	7,9 ± 0,100	1	8 ± 0	1	1
Monks-3	1	1	1	5 ± 0	1	5 ± 0	1	5 ± 0	1	1
BCW	0,957	0,995	0,958	12,9 ± 0,277	0,905	7,6 ± 0,4	0,947	9,4 ± 0,542	1	0,995

Table 4. The results GARulExNN in term of the *fidelity* rate. (The sign " –" standing in the cell denotes the lack of information)

Data set	***Primary set of rules (EA_1)***		***Rules obtained 'after the last generation' (EA_2)***		***Rules obtained as 'the best individual during evolution' (EA_2)***		***The best found set of rules (EA_2)***	
	Predictive fidelity	*Number of rules*	*Predictive fidelity*	*Number of rules*	*Predictive fidelity*	*Number of rules*	*Predictive fidelity*	*Fidelity*
Iris	0,980	8,9 ± 0,407	0,979	5,8 ± 0,389	0,980	7,5 ± 0,453	1	1
Wine	0,947	14 ± 0,537	0,947	9,8 ± 0,389	0,946	11,3 ± 0,517	0,941	0,994
Monks-1	1	8,3 ± 0,153	1	7,9 ± 0,100	1	8 ± 0	1	1
Monks-3	1	5 ± 0	1	5 ± 0	1	5 ± 0	1	1
BCW	0,966	12,9 ± 0,277	0,987	7,6 ± 0,4	0,966	9,4 ± 0,542	1	1

example (that is the same as the neural network) or do not classify it at all.

Table 6 shows comparison with other extraction methods respectively with the same data set as they were described in the literature. For the data sets: *Iris, Wine, Monks-1* GARulExNN outperformed the results of Santos's method. Other methods were tested with *Breast Cancer Wisconsin* data set, where the results obtained by GARulExNN were a little bit worse than those obtained by AntMiner but this comparison is not adequate to the whole extent. AntMiner acquires rules from the data while GARulExNN from neural network, what can introduce an additional disruption in the classification process.

5. Conclusions

In this paper the method of the rule extraction from neural networks via hierarchical evolutionary algorithm is presented. Its performance is evaluated on five public domain data sets. Our computational results show that GARulExNN can produce a set of rules describing the behaviour of the neural network with a good rate of *accuracy*. In the domain of rule extraction from neural networks 'good accuracy rate' denotes how good the set of rules can mimic the actions of the network. During our

Table. 5. The example of the extracted set of rules

EA_1	1	IF PetalLen IN (1,1339949; 2,5258226) THEN Setosa	*Predictive accuracy=1 Accuracy=1 Predictive fidelity= 1 Fidelity=1*
	2	IF PetalLen IN (2,8424874; 5,0500904) AND PetalW IN (0,8772328; 1,4421928) THEN Versicolour	
	3	IF PetalLen IN (4,9506105; 6,7152651) THEN Virginica	
	4	IF SepalLen IN (4,5179296; 4,87897) THEN Setosa	
	5	IF PetalLen IN (2,073092; 4,8667007) AND PetalW IN (0,4399408; 1,6047208) THEN Versicolour	
	6	IF PetalW IN (1,6293832; 2,4630256) THEN Virginica	
	7	IF PetalLen IN (1,0502739; 2,5867165) THEN Setosa	
	8	IF SepalW IN (3,052112; 4,2694208) AND PetalW IN (0,7661368; 1,766152) THEN Versicolour	
	9	IF SepalW IN (2,3748848; 2,6995472) AND PetalW IN (1,450648; 2,482096) THEN Virginica	
EA_2 ('the best during evolution')	3	IF PetalLen IN (4,9981114; 6,7152651) THEN Virginica	*Predictive accuracy=1 Accuracy=1 Predictive fidelity=1 Fidelity=1*
	4	IF SepalLen IN (4,5179296; 4,87897) THEN Setosa	
	5	IF PetalLen IN (2,413522; 4,8667007) AND PetalW IN (0,4399408; 1,6047208) THEN Versicolour	
	6	IF PetalW IN (1,6293832; 2,4630256) THEN Virginica	
	7	IF PetalLen IN (1,0502739; 2,5867165) THEN Setosa	
	8	IF SepalW IN (3,052112; 4,2694208) AND PetalW IN (0,7661368; 1,766152) THEN Versicolour	
	9	IF SepalW IN (2,3748848; 2,6995472) AND PetalW IN (1,450648; 2,482096) THEN Virginica	

Table 6. The Comparison of the GARulExNN results with other methods

Data set	***GARulExNN*** *('the best sets of rules during evolution' EA_2)*		***Santos, Nievola, Freitas (GA)[7]***		***AntMiner algorithm[5]***		***PRIM algorithm[10]***	
	Predictive accuracy	*Number of rules*	*Predictive accuracy*	*Number of rules*	*Predictive accuracy*	*Number of rules*	*Predictive accuracy*	*Number of rules*
Iris	0,960	7,5 ± 0,453	0,933	10,6	–	–	–	–
Wine	0,882	11,3 ± 0,517	0,853	88,6	–	–	–	–
Monks-1	1	8 ± 0	0,998	34,2	–	–	–	–
Breast Cancer Wisconsin	0,947	9,4 ± 0,542	–	–	0,955	5,60 ± 0,80	0,946	7

research it became obvious that even evolutionary algorithm EA_1 can produce a set of rules with a high *predictive accuracy* rate but this comes with a cost of increased complexity of the rule set itself.

Therefore the main goal of the upper level algorithm was to 'prune' the primary set of rules. The results have shown that this was achieved and the final sets of rules are composed of less number of rules but this causes decreasing of the *accuracy* of such rule classifier. However, this is not a major loss because the *fidelity* rate of the final set of rules is still stable. As it was mentioned in previous section this means that the set of rules classifier tends to correctly classify an example or not to classify it at all rather than to give a false classification. This behaviour may be desirable in the medical domains, where it is safer to admit that we do not know what sort of illness the patient has than to cure him of disease that he does not have.

The weakness of our method is the amount of parameters that should be initially tested in order to find the best final set of rules. The purpose of EA_2 was to find set of rules with a small number of rules preserving the *accuracy* obtained by EA_1. So as a result EA_2 finds more general set of rules but the *accuracy* is decreased. Therefore we decided to present two final sets of rules: extracted after the last generation and the set of rules stored as the best set found during the evolution on upper level. Usually that set was found early in the evolution process and was very similar to the primary set of rules.

References

[1] Andrews R.: Survey and critique of techniques for extracting rules from trained artificial neural networks (Knowledge-Based Systems, Volume: 8, Issue: 6, December, 1995, pp. 373-389)

[2] Arbatli A. D., Akin, H. L.: Rule extraction from trained neural networks using genetic algorithms Nonlinear Analysis, Volume: 30, Part 3, December, 1997, pp. 1639-1648

[3] Blake C. C., Merz C.: UCI Repository of Machine Learning Databases, University of California, Irvine, Dept. of Information and Computer Sciences, 1998

[4] Darbari A.: Rule Extraction from Trained ANN:A survey, Technical report Institut of Artificial intelligence, Dep. of Comp. Science, TU Dresden, 2000

[5] Parpinelli R., Lopes H., Freitas A.: An Ant Colony Based System for Data Mining: Applications to Medical Data, in: H. Abbass, R. Saker, C. Newton. (eds.) Data Mining: a heuristic approach pp.191-208. London: Idea Group Publishing 2001;

[6] Radcliffe N. J., Surry P. D.: Co-operation through Hierarchical Competition in Genetic Data Mining, Technical Report EPC-TR94-09, Edinburgh Parallel Computing Centre, 1994,

[7] Santos R., Nievola J., Freitas A.: Extracting Comprehensible Rules from Neural Networks via Genetic Algorithm, In Proc.2000 IEEE Symp. On Combination of Evolutionary Combination and Neural Network, pp. 130-139, San Antonio, RX, USA, 2000

[8] Sethi I.K., Yoo J. H.: Symbolic Approximation of Feedforward Neural Networks in E.S. Gelsema and L.N.Kanal (Eds.) Pattern Recognition in Practice, pp. 313-324, North-Holland, 1994.

[9] Setiono R.: Extracting rules from pruned neural networks for breast cancer diagnosis, Artificial Intelligence in Medicine, Volume: 8, Issue: 1, February, 1996, pp. 37-51

[10] Silva, R. B. A. Ludermir, T.B.: Neural Network Methods for Rule Induction, 1999; Proceedings of the 1999 International Coinference on Neural Networks, Paper Paper 2099, Washington - DC

[11] Taha I., Ghosh J.: Symbolic Interpretation of Artificial Neural Networks, Tech. Rep. TR-97-01-106, The Comp. and Vision Research Center, Univ. of Texas, Austin, 1996.

[12] Thrun S. B.: Extracting provably correct rules from artificial neural network, Technical Report, Institut für Informatik, Universität Bonn, Bonn,1994.

[13] Towell, G. G., Shavlik, J.: Extracting refined rules from knowledge based neural networks. Machine Learning vol. 131 (1993), 71–101