Rule Extraction from Neural Network by Genetic Algorithm with Pareto Optimization

Urszula Markowska-Kaczmar and Paweł Wnuk-Lipiński

Wroclaw University of Technology, Poland
kaczmar@ci.pwr.wroc.pl

Abstract. The method of rule extraction from a neural network based on the genetic approach with Pareto optimization is presented in the paper. The idea of Pareto optimization is shortly described and the details of developed method such as fitness function, genetic operators and the structure of chromosome are shown. The method was tested with well known benchmark data sets. The results of these experiments are presented and discussed.

1 Introduction

In some application of neural networks (NN) the fundamental obstacle is the trouble with understanding the method, in which the solution is produced by NN. This is the reason to develop methods extracting knowledge from NN in the comprehensible way [3], [2], [4]. Often it has the form of propositional rules. In the last few years many rule extraction methods were developed. The expressive power of extracted rules varies depending on the method. Many of them are dedicated to problems with binary attributes [6] or they need special training rule [5], so the need for developing an efficient method of rule extraction still exists.

In the paper we describe new rule extraction method based on evolutionary algorithm and Pareto optimization, called $GenPar$[1]. Genetic algorithm (GA) is the well known technique for searching single optimal solution in a huge space of possible solutions. In the case of rule extraction we are interested in acquiring the set of rules and satisfying different criteria. This possibility offers multiobjective optimization in Pareto sense [7].

2 Multiobjective Optimization in Pareto Sense

A a lot of problems require to make allowances for different objectives $f_i (i = 1, .., k)$. Sometimes they are mutually exclusive. In most cases there may exist several comparable solutions representing different trade-offs between objectives. In the Pareto approach each individual is evaluated for each single objective f_i. The quality of the specific solution is expressed by the objective vector. The

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

L. Rutkowski et al. (Eds.): ICAISC 2004, LNAI 3070, pp. 450–455, 2004.

solutions can be categorized as *dominated* or *nondominated*. The solution **a** is *dominated* if there exists other solution **b** and the following is satisfied: $f_i(\mathbf{a}) \leq f_i(\mathbf{b})$ for each $1 \leq i \leq k$. If the solution is not dominated by other solution it is called *nondominated* or Pareto-optimal [7]. It represents the solution, which is the best for all objectives. The problem is how to find *nondominated* vectors. One of the possible approach is its calculation as a product of components f_i of the objective vector, which is implemented in *GenPar*.

3 Basic Concepts of the *GenPar* Method

GenPar is the method of a NN description in classification problems by means of genetic algorithm. For that reason NN produces the training examples for developed method. In other word, it is used as an oracle for the proposed rule extraction method. General idea of *GenPar* in pseudocode is presented in Table 1. It can be easily noticed that the evaluation of one individual (set of rules) needs a cycle, which is composed of some steps. At the beginning, the chromosome is decoded to a rule set. Afterwards, the set of training patterns are applied to the rule set and NN. Each individual is evaluated on the base of accuracy (interpreted as fidelity decreased by the number of misclassified examples) and comprehensibility (expressed by the number of rules and the number of premises). Then, the algorithm searches for nondominated individuals and calculates the global adaptation value for each of them. In the last step individuals are drawn to the reproduction and finally by applying genetic operators the new population is produced.

3.1 The Form of Chromosome

Each rule has IF – THEN form. The body of the rule is a conjunction of premises. Each premise imposes a constraint on the values of attribute given on the neural network input. After THEN stands a class label (code) and it forms the conclusion part of the rule. For the real type of attribute a_i the premise shows the range of values $[a_{imin}; a_{imax}]$, for which the rule is active. For attribute b_j of enumerative type premise contains the subset B_j of values, for which rule can fire.

In the *GenPar* the hierarchic way of coding is applied. On the first level there is a list of genes, which codes the set rules (Fig.1). The second level is created by genes representing single rules. That form of an individual produces the solution directly by decoding a chromosome but it increases the complexity of the chromosome. Even so, we implemented the chromosome in that form because GA is very fast searching wide spaces of solutions and the perspective to obtain the set of rules directly from one chromosome seems very promising. On the second level a code of a single rule is composed of genes representing premises (constraints set upon attributes) and one gene of conclusion. Before each gene representing real type attribute stands flag (A). It can take two values 0 or 1. Only when it is set to value 1 the premise is active. The flag is followed by two limits of a range $[a_{imin}; a_{imax}]$. Enumerate parameters are coded in the bit sequence form, where

Table 1. Pseudocode of the *Genpar* method

```
For every individual in the population do
    Decode genotype in the rule set
    For every pattern do
        Calculate the neural network response
        Calculate the rule set response
        Compare both responses
    End {For}
    Calculate the accuracy part of the fitness function
    Calculate the comprehensibility part of fitness function
End {For}
While in the base population exist individuals do
    Find nondominated individuals in the base population
    Set nondominated individual's fitness as:
            1/(number of nondominated individuals +
                    number of earlier evaluated individuals)
    move evaluated individuals to the temporary population
End {While}
base population := temporary population
While size of the offspring population is less than size of parent's population do
    Draw two parents from parent's population
     Make offspring from the drawn parents
    Add that offspring to the offspring population
End {while}
the offspring population becomes the base population
```

for every possible value of the attribute stands one bit. If a bit is set to 1, rule is active for that value of attribute. Binary attribute is treated as enumerative one.

3.2 Genetic Operators

In the *GenPar* two genetic operations are implemented. That is crossover and mutation. Because the individual is a list of genes with variable length, the operators differ from their standard form. Individuals are selected by applying roulette wheel method. Because of individuals have different genotype length, we have to assure that both parents have equal gene contribution in the offspring chromosome. During the crossover, the offspring chromosome is created by a

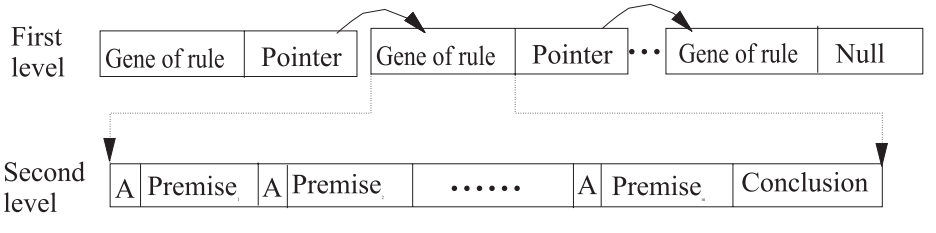

Fig. 1. The form of chromosome in the *GenPar*

random choice of a gene from one of parents according to (1), till to the length of shorter parent.

$$Y_i = \begin{cases} X_i^1 & \text{when} \quad \mu < 1 - \frac{r_2}{r_1 + r_2} \\ X_i^2 & \text{in other case} \end{cases} \tag{1}$$

In (1) X_i^1 is i-th gene of the first parent; X_i^2 is i-th gene of the second parent; Y_i is i-th gene of offspring; r_1 is the number of genes coding rules of the first parent; r_2 is the number of genes coding rules of the second parent; μ is a random number between 0 and 1. Genes of longer parent that do not posses their counterparts at the second parent are copied to the offspring chromosomes with the probability p given in (2), where the meaning of symbols is defined by (1).

$$p = 1 - \frac{r_2}{r_1 + r_2} \tag{2}$$

There are two forms of mutation. The first one, on the higher level, adds or deletes a gene representing a single rule. On the second level it is realized as a random change of binary values like: flag or a bit in the binary sequence for enumerative parameters. Limit values of real attributes, which are implemented as 32 bits numbers, are mutated in the same way.

3.3 Fitness Function

Individuals are evaluated because of two objectives: *accuracy* and *comprehensibility*. Because of Pareto optimization, for both objectives single evaluation function is created. The first one called *accuracy* expresses the $fidelity$ between classification based on the set of rules and that one made by NN for every pattern, which is decreased by the number misclassified patterns (3).

$$accuracy = fidelity - misclassified \tag{3}$$

In other words, *fidelity* in (3) is the number of properly classified patterns by the set of rules, while *misclassified* is the number of misclassified patterns by the set of rules (in the sense of classification made by NN). *Comprehensibility* of the set of rules is described by (4). It depends on the rule number (*nrules*) in the individual and the number of premises (*npremises*) in rules.

$$comprehensibility = \frac{x}{x + (nrules + npremises)} \tag{4}$$

Parameter x enables the user to determine, which criterion is more important. For large value of x, the number of rules and premises is not significant, so the *comprehensibility* is not important in the final fitness value. For a small value of x, *comprehensibility* is near to $x/(rulenumber + premisenumber)$ and that objective will dominate. In the Pareto optimization in order to calculate the global fitness value nondominated individuals have to be found. In *GenPar* it is implemented by means of product of the *accuracy* and *comprehensibility*

adaptation values. Next, nondominated individuals get the highest global fitness value and they are moved to the temporary population. After that, from remaining individuals nondominated individuals have to be found in the new base population. They get a little bit smaller value of the global fitness value than the former nondominated individuals. That cycle is repeated until the base population will be empty. In the next step, the temporary population becomes the new base population.

4 Experiments

The experimental study has the aim to test whether the proposed method works effectively independently on the type of attributes. All presented further experiments were performed with multilayered feedforward neural network, with one hidden layer, trained by backpropagation. In experiments we use *Iris*, *Monk*, LED and *Quadruped Mammals* benchmark data sets included in [1]. The characteristics of these data sets and the classification level of neural network are shown in Table 2. In all presented experiments population consists of 30 individuals. Each individual in the initial population is created by the choice of 20 rules in random. On the base of initial experiments the probability of mutation was set to 5 %. The results of experiments are shown in Table 3, where for different values of x the results are the round averages from 5 runs of program.

It is easy to notice that for the small value of the parameter x (*Iris* data set), *GenPar* found the most general rule, which properly classified all 50 examples from one class. With the increase of the value x, grew the number of properly classified examples (*fidelity*). These observation in experimental way confirm the appropriate influence of the parameter x. The *fidelity* of *GenPar* is about 96 %, what is comparable with the results of other methods (for example 97 % for FullRe [6]), but it offers the user the easy way to decide which objective is the most important one. *Monk Problems* were chosen in order to test efficiency of the method for enumerative type of attributes. The results are not included in Table 3, but for *Monk-1* and *Monk-3*, the method easily found the set of appropriate rules (respectively 6 and 2 rules). The most difficult problem was *Monk-2*. It is because of the training examples were created on the base of the relationship between two input attributes what was not detracted in the form of rule assumed in the *GenPar*. The data set *Mammals* was used in order to test scalability of the method. In this experiment population consists of 50 individuals. Each of them initially contained 30 rules. Finally, only 4 rules were found that have fidelity equal to 90 %. In the last experiment LED problem was used. LED-7-1000 in Table 3 is the abbreviation for 7 input attributes and 1000 noisy examples. Thanks of the NN ability to remove noise it was much easer to find rules for this difficult LED problem than directly from data.

5 Conclusion

Even the *GenPar* was tested with the feedforward NN it is independent on the NN architecture. There is no need for special learning rule during rule extraction,

Table 2. Characteristic of data sets used in experiments

problem	attributes	classes	examples	NN classification in %
Iris	4	3	150	98
Monk	5	2	124	100
LED-7-100	7	10	100	89
LED-7-1000	7	10	1000	67
LED-24-1000	24	10	1000	69
Mammals	72	4	500	97

Table 3. Results of experiments for *GenPar*

Iris			LED- 7-1000			Mammals		
x	n. of rules	fidelity	x	n. of rules	fidelity	x	n. of rules	fidelity
100	1	50	100	1	87	1000	2	182
500	2	94	500	4	324	5000	4	468
1000	2	94	1000	5	473	10000	4	471
2000	4	137	2000	13	918			
5000	5	144	5000	13	928			

as well. Experimental study has shown that it works efficiently for both continuous and enumerate attributes. Thanks to Pareto optimization it easy for the user to indicate, which criterion is more important for him. Developed method treats the neural network as a black box. It means that it uses NN to produce training example for the rule extraction method. *GenPar* can be easily used for extracting set of rules directly from data. However because of the good ability of NN to remove noise its using is recommended.

References

1. Blake C. C., Merz C.: UCI Repository of Machine Learning Databases, University of California, Irvine, Dept. of Information and Computer Sciences (1998)
2. Darbari A.: Rule Extraction from Trained ANN:A survey, Technical report Institut of Artificial intelligence, Dep. of Comp. Science, TU Dresden (2000)
3. Mitra S., Hayashi Y.: Neuro-fuzzy rule generation: Survey in soft computing framework, IEEE Transaction on Neural Networks, (2000).
4. Santos R., Nievola J., Freitas A.: Extracting Comprehensible Rules from Neural Networks via Genetic Algorithm, Proc.2000 IEEE Symp. On Combination of Evolutionary Algorithm and Neural Network (2000) pp. 130-139, S. Antonio, RX, USA
5. Setiono R.: Extracting rules from pruned neural networks for breast cancer diagnosis, Artificial Intelligence in Medicine, Vol: 8, Issue: 1, Feb. (1996) pp. 37-51
6. Taha I., Ghosh J.: Symbolic Interpretation of Artificial Neural Networks, Technical Rep. TR-97-01-106, University of Texas, Austin (1996)
7. Zitzler E, Thoele L.: An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach http://citeseer.nj.nec.com/225338.html