Genetic Algorithm Optimization of Knowledge Extraction from Neural Networks

Vasile Palade, Gheorghe Negoita, Viorel Ariton

Department of Applied Informatics, "Dunarea de Jos" University of Galati, Romania.
Email:{pvasile, gnegoita, vio}@cs.ugal.ro

Abstract

Neural networks have been criticized for their lack of human comprehensibility. First, this paper proposes an extraction method of crisp if-then rules from ordinary Backpropagation neural networks. Then, the paper presents a mechanism that compile a neural network into an equivalent set of fuzzy rules. Genetic algorithms are used to find the right structure of the fuzzy model equivalent with the neural network, and then to find the best shape of the membership functions. In order to reduce the number of fuzzy rules when wish to compile a neural network with many inputs, genetic algorithms are used to find the best hierarchical structure of the fuzzy rules, considering the relations between the network inputs.

1. Introduction

Neural networks have been applied, with remarkable success, to a variety of real-word problems: control field, speech recognition, medical diagnosis, image computing etc. The major shortcoming of neural networks is represented by their low degree of human comprehensibility. Many authors have focused on solving this shortcoming of neural networks, by compiling the knowledge captured in the topology and weight matrix of a neural network, into a symbolic form; most of them into sets of ordinary if - then rules (Towell and Shavlik, 1993; Yoo, 1993; Thrun, 1994), others into formulas from propositional logic or from nonmonotonic logics (Pinkas,1992), or into sets of fuzzy rules (Kosko, 1992; Palade et al., 1998). The fuzzy neural network are often used as an auto-tuning method for the determination and the adjustment of fuzzy rules (Lin and Lee, 1991).

In the paper, we developed a mechanism for compiling the knowledge captured in the network's weight matrix, first into a set of crisp if-then rules, and then into a set of fuzzy rules. The method for rule extraction developed in section 2 of the paper can be used to check the compatibility of a rule with the network, and to obtain a set of crisp if-then rules out of a weight matrix of a neural networks. The method presented in the second part of the paper (section 3 and 4) can be used to develop a set of fuzzy rules from experimental data, without any assistance from a human expert, in the field of control or pattern recognition. Our system can improve neural network based systems, such as neural controllers, with explanation facilities. For example, the decisions taken by a neural controller can be presented to the user in the form of fuzzy rules, leading to an increase of confidence in the controller action. First, genetic algorithms are used for finding the best hierarchical structure of the fuzzy rules, and after that for tuning the shapes and the number of membership functions. The method of interval propagation is used to extract a set of traditional if-then rules. This set of rules is used to obtain the initial population of solutions, given a good point of start in searching the right number and appropriate shapes of the membership functions. Section 4 shows how to find a hierarchical structure of the fuzzy rule set extracted from a neural network, using also genetic algorithms. Our goal is to minimize the difference between the neural network and the fuzzy system obtained after finding the best hierarchical structure and membership function tuning.

2.Rule Extraction by Interval Propagation

In this section, we present a method for rule extraction from neural networks with continuous inputs and outputs. We named our method, the *method of interval propagation*. The rules extracted by our method are crisp if – then rules, in the following form:

$$\textit{if}\ \ (a_1 \le x_1 \le b_1)\ \text{and}\ (a_2 \le x_2 \le b_2)\ \dots\ \text{and}\ (a_m \le x_m \le b_m)$$

then $c_j \le y_j \le d_j$

where $x_1, x_2, \dots, x_m$ are the inputs of the network and y_j is the j output of the network, $j=1,2,\dots,n$.

Another method which tries to extract rules in the same form, out of a trained neural network, is the VIA method, developed by Thrun in (Thrun, 1994). VIA method refines the intervals of all units in the network, layer by layer, by techniques of linear programming, such as Simplex algorithm, propagating the constraints forward and backward through the network. The problem is that VIA method may fail sometimes to decide if a rule is compatible or not with the network. Also the intervals obtained by VIA method are not always optimal. Our method eliminates these drawbacks of the VIA method.

Given P a layer in the network, and S the next layer. Every node in layer S calculates the value $x_i = f(\sum_{k\in P} w_{ik}x_k + \theta_i)$, where x_k is the output (activation value) of node k in layer P, x_i the output of node i in the layer S, w_{ik} the weight of the link between node k in layer P and node i in layer S, θ_i the bias of node i, and f the transfer function of the units in the network. We can write the relations:

$$(\forall)\, k \in P \qquad x_k \in [a_k;b_k]$$
$$(\forall)\, i \in S \qquad x_i' = \sum_{k\in P} w_{ik}x_k + \theta_i$$
$$x_i = f(x_i')$$

For every node $i \in S$, we note with w_{il}^+, $l \in P_i^+$, the positive weights, and with w_{il}^-, $l \in P_i^-$, the negative weights. ($P_i^+ \cup P_i^- = P$). The interval of variation for x_i', $[a_i';b_i']$, $(\forall) i \in S$, is determined in the following way:

$$a_i' = \sum_{l\in P_i^+} w_{il}^+ a_l + \sum_{r\in P_i^-} w_{ir}^- b_r + \theta_i$$

$$b_i' = \sum_{l\in P_i^+} w_{il}^+ b_l + \sum_{r\in P_i^-} w_{ir}^- a_r + \theta_i$$

and the variation interval for the activation value x_i of node i in layer S is $[a_i;b_i]$, where: $a_i = f(a_i')$ and $b_i = f(b_i')$.

In this way, the intervals are propagated, layer by layer, from the input layer to the output layer. So, given the variation intervals for inputs, the intervals of variation for outputs are determined. This is the forward phase. Some of the inputs may be unconstrained, and in this case the intervals are propagated forward across the network layers, assigning the interval of maximum variation ([0; 1]) for unconstrained inputs.

The backward phase appears when it is given the interval of variation for output and eventually for some inputs, and it must be determined the interval for unconstrained inputs. Suppose $x_1, x_2, \dots, x_k$ are the constrained inputs after renumerotation, and $x_{k+1}, \dots, x_m$ the unconstrained inputs, and we want to determine rules when $(a_1 \le x_1 \le b_1)$ and ...

... $(a_k \le x_k \le b_k)$ and $(c_j \le y_j \le d_j)$.

First, it is checked the compatibility of the following rule:

if $(a_1 \le x_1 \le b_1)$ and $(a_2 \le x_2 \le b_2)$ ....
and $(a_k \le x_k \le b_k)$
then $c_j \le y_j \le d_j$

with the network, assigning the maximum interval ([0; 1]) for unconstrained inputs. By forward propagation, the variation interval $[c_j';d_j']$ for output is determined. If $[c_j';d_j'] \subset [c_j;d_j]$, then the rule given above is a general rule, and it does not have sense to look for the variation intervals of remained inputs. If the intersection $[c_j';d_j'] \cap [c_j;d_j]$ is empty set, then the rule is incompatible with the network. Otherwise, be $y_j^* \in [c_j';d_j'] \cap [c_j;d_j]$.

By inverting the neural network (Scott, 1993; Palade, 1999) it is determined the input $x^*=(x_1^*, x_2^*, \dots x_m^*)$ of the network which produce the output y_j^*. The idea is to find the maximal intervals around the values x_l, $l=k+1, \ldots, m$, so that the corresponding rule to be compatible with the network. For example, beginning with input x_l, the right margin b_l of the variation interval is established to $b_l = x_l^* + \frac{1 - x_l^*}{2}$.

If the rule with $x_l \in [x_l^*; b_l]$ is compatible with the network, then the interval is enlarged, otherwise is shrinking, with a technique of dividing intervals into two halves, until the right margin b_l and a_l are determined with a given error. The procedure continues until all the variation intervals for all unconstrained inputs are determined. The hypercubs determined at the input depend on the start position – x*, and on the order of the determination of the variation intervals for unconstrained inputs.

3. Mapping Neural Networks into Fuzzy Rules

In this section, we present a method for mapping a neural network into an equivalent set of fuzzy rules. Genetic algorithms are used for tuning the shape, the position, and the number of membership functions. In the algorithm, it is used the method of interval propagation presented in the previous section, to initialize the population of solutions and to give a good start in search, when using trapezoidal membership functions.

The algorithm for mapping a neural network into a set of fuzzy rules is summarized bellow:

(1) Set a number of certain membership functions, so that they divide the input space equally.
(2) For the upper bases of the membership functions of the inputs, calculates, by method of interval propagation, the corresponding intervals of the output.
(3) Based on the crisp rules obtained in the previous step, construct an initial population of solutions, initializing randomly the parameters from the lower bases of trapezoidal membership functions.
(4) Tune the shape of membership functions, until the best fitness value becomes less than the target one, or searching time reaches the limited generations. If the limited number of generations is reached, go to the next step (we have to find another structure of the fuzzy model). Otherwise, the minimal and optimal fuzzy model is obtained.
(5) Let the mutation operator to prune a membership function in one of the inputs or in the output. Go to step number 2.

For the beginning, the chromosomes can be lists of real numbers which represent in order the parameters of the membership functions both for inputs and output. The algorithm was given for trapezoidal membership functions, but it can be adapted for triangular and gaussian membership functions. What differs, is the way of obtaining the initial population of likely solutions. For every combination of the input linguistic terms, the idea is to determine the membership function of an output linguistic term in the consequence of the rule. The center (when using triangular and gaussian membership functions) or the upper base (when using trapezoidal membership functions) of the membership function of the output linguistic term is determined using a crisp rule.

The disadvantage of this method is the exponential increasing of the number of output linguistic terms, which may cause losing the linguistic significance for the output terms (for example Small, Medium, Big a.s.o.). If every input x_i is described by β_i linguistic terms, taking into consideration all the combinations, it will result $\prod_{i=1}^{m} \beta_i$ output linguistic terms. To avoid this problem, one aproach is to let the genetic algorithms to prune some of the output linguistic terms. The fitness function will be $f=E+\alpha L$, where E is the sum of squared error between the output of the neural network and the output of fuzzy system represented by the current solution in the population, L is the number of output linguistic terms, and α a coefficient.

The structure of the chromosomes must be modified, in order to eliminate some of linguistic terms. Every chromosome contains three fields. The field NFT (number of fuzzy terms) is formed by an integer which codes the number of output linguistic terms. The field PMF (parameters of membership functions) codes the parameters of membership functions, both for inputs and output. The field SFR (structure of fuzzy rules) contains a list of $\prod_{i=1}^{m}\beta_i$ integers, each of them representing the number of output linguistic term in the consequent of a rule, from the $\prod_{i=1}^{m}\beta_i$ possible rules. The crossover and mutation have to be able to work with this kind of chromosomes. The mutation is the operator which will determine the pruning of some output linguistic terms. In order to let the mutation to prune some of the linguistic terms from the input, the NFT field must be modified, so that to code the number of linguistic terms for each input. Also, if not wish to consider all the conjunctive rules given by all the combinations of input linguistic terms, the SFR field must be modified to represent the structure of the considered rules.

The problem of exponential increasing of output linguistic terms appears just in the case we want to determine a Mamdani fuzzy model equivalent with the network (when in the consequent of the rules we have membership function). The problem does not have sense when the fuzzy model equivalent with the network is a Sugeno fuzzy model of order zero, with singletons in the consequent of the rules. In this case the chromosomes will code only the parameters of membership functions of the inputs. The value in the consequent of a rule is determined by calculating the output of the network when the inputs of the network take the values corresponding to the centers of the membership functions from the antecedent of the rule.

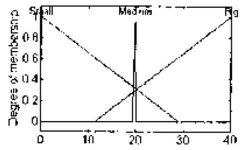
Example: In order to test the abilities of GAs to find the best membership functions, it was taken a neural network which represents the neural controller for a common benchmark process (Scott, 1993), the water tank, described by the following equation:

$$\frac{dy}{dt}=\frac{1}{A}[u-K\sqrt{y}]$$

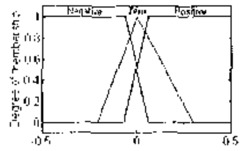
where K=7, A=30, $u\in[0;40]$, $y\in[0;10]$. We intend to replace the neural controller with an equivalent fuzzy controller, with the method described in this section. The neural network which represents the controller had one input – ε (the difference between the output of the system y and the setpoint y_d) and one output – u (the command, which becomes the input to the process). From the trained neural network we extracted the following three crisp rules:

if	$\varepsilon \geq 0.06$	*then*	$u \geq 40$
if	$\varepsilon = 0$	*then*	$u = 19.8$
if	$\varepsilon \leq -0.06$	*then*	$u \leq 0$

The first and the third rule were obtained by interval propagation method, the second were obtained simply calculating the output of the neural network for ε=0. The initial population is obtained initializing randomly the parameters of



membership functions from the lower bases, and the others parameters of membership functions with the values taken from the three rules given above. After 200 generations, we obtained the membership functions depicted in figure 1.



a) b)

Figure 1. Obtained membership functions for controller input ε (a) and output u (b)

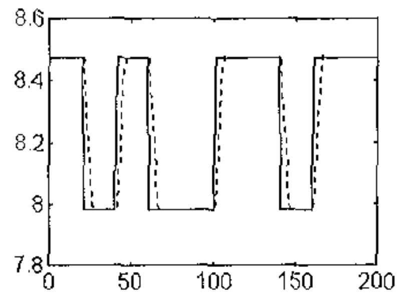


Figure 2. The behaviour of fuzzy controller to setpoint changes (dotted – process output, continuous – setpoint)

The population had 100 chromosomes, the mutation rate was 4%. The fuzzy rules which describe the fuzzy controller are the following:

if	ε is Negative	*then*	u is Small
if	ε is Zero	*then*	u is Medium
if	ε is Positive	*then*	u is Big

and the performances of the controller regarding the setpoint changes are shown in figure 2. The three fuzzy rules translates in a human understandable form the behaviour of the neural controller.

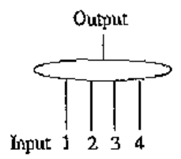
4. Hierarchical Aproach

The main problem encountered by our method is the exponential increasing of the number of fuzzy rules with the increase of the dimension of the input space. The number of rules is the product of the number of membership functions for each input. In order to reduce the number of membership functions and consequently the number of fuzzy rules, we applied GAs to find a hierarchical structure of the fuzzy system, considering the relations between the input variables, in a similar manner as Shimojima et al. used in their work (Shimojima et al., 1995).

The key idea is to collect the inputs which have closer relationships in one fuzzy unit. For example, let's take a network with 4 inputs and one output, shown in figure 3, and let's consider three equidistant membership functions in each input space. The total number of fuzzy rules which we could obtain is 81 = 3 x 3 x 3 x 3, considering all the combinations. Now, let's suppose there are some relationships between the inputs 1 and 2, on one hand, and between the inputs 3 and 4, on the other hand. In this case, the neural network can be described by the hierarchical fuzzy system shown in figure 4. Every fuzzy unit is described by 3x3 rules. Then, the total number of fuzzy rules which describe the network is drastic reduced.

The coding method:

If the network has n inputs, we arrange 2^{n-2} units in a tree with 1, 2, 4, 8,... units on each level. Every input is assigned to a fuzzy unit in the tree. If a fuzzy unit has only one input, the input is passed to the upper fuzzy unit in the tree. For example, if n = 5, we arrange the units in the tree shown in figure 5. With the assigning given in figure 5, and after eliminating unnecessary fuzzy units, we obtain the structure given in figure 6.



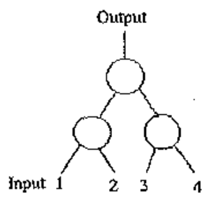


Figure 3. Figure 4.

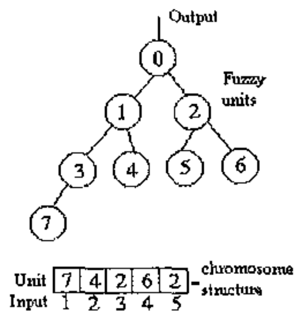


Figure 5. The coding method

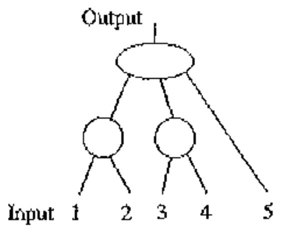


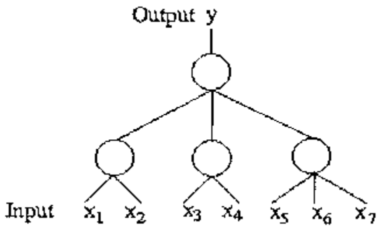
Figure 6. The corresponding hierarchical structure

The structure of a chromosome is shown in figure 5. A gene corresponds to a network input and contains the number of the fuzzy unit which is assigned to that input. As crossover operator we used a common one point crossover operator. The fitness function used for this part is $f = E + \alpha * N$, where E is the sum of the squared error between the network output and the output of the equivalent fuzzy model, N is the number of fuzzy rules, and α a coefficient. In fact, the coding method is the same used by Shimojima et al. (Shimojima et al., 1995), instead we used another fitness function.

As an example, we tried to find a hierarchical structure for a neural network which implements the function: $f(x_1,x_2,x_3,x_4) = (\min(x_1,x_2) + \min(x_3,x_4)+\max(x_5,x_6,x_7))/7$. The initial population contained 40 solutions. Meeting our expectations, the best hierarchical structure which we could obtain is shown in figure 7. This was happened in around 50 generations. For the determination of the best hierarchical structure, we used Sugeno fuzzy rules with singletons in the consequents. Every fuzzy unit is described by a number of such fuzzy rules.

Figure 7. The obtained hierarchical structure

In conclusion, the algorithm for mapping a neural network into a set of fuzzy rules is



divided into two main subprocedures:

1. Find the best hierarchical structure of the fuzzy model equivalent with the network.
2. Tune the shapes of the membership functions and the number of linguistic terms of the hierarchical structure found in the previous step.

5. Conclusion

A method of rule extraction from neural networks with continuous inputs and outputs was developed in this paper. The method was called method of interval propagation. Also, we improved our algorithm for compiling neural networks into an equivalent set of rules, based on a hierarchical aproach, in order to address the

networks of big sizes. A future research idea is to try to convert a neural network into a set of Sugeno fuzzy rules of order bigger than zero, to reduce the dimension of the fuzzy model equivalent with the neural network.

References

Ishigami H., T. Fukuda, T. Shibata and F. Arai (1995). Structure Optimization of Fuzzy Neural Network by Genetic Algorithm. *Fuzzy Sets and System* 71, pp. 257-264.

Kosko B. (1992). *Neural Networks and Fuzzy Systems.* Prentice - Hall International Inc.

Lin C.T. and C.S. George Lee (1991). Neural - Network Based Fuzzy Logic Control and Decision System. *IEEE Transactions on Computers*, vol. 40 No. 12, 1991, pp. 1320-1336.

Palade V. (1998). *Hybrid Expert Systems for Process Control.* PhD thesis, "Dunarea de Jos" University of Galati, dec. 1998.

Palade V., S. Bumbaru and M. Negoita (1998). A method for compiling neural networks into fuzzy rules using genetic algorithms and hierarchical aproach, *Proc. of 2nd Int. Conf. on Knowledge-Based Intelligent Electronic Systems*, vol 2 pp. 353-358, Adelaide-Australia.

Pinkas G. (1992). *Logical Inference in Symmetric connectionist Networks.* PhD. thesis, Washington University.

Scott G. (1993). *Knowledge – based artificial neural networks for process modeling and control.* PhD thesis, University of Wisconsin – USA.

Shimojima K., T. Fukuda and Y. Hasegawa (1995). Self-tuning Fuzzy Modelling with Adaptive Membership Function, Rules, and Hierarchical Structure Based on Genetic Algorithm. *Fuzzy Sets and Systems* 71, pp. 295-309.

Thrun S.B. (1994). *Extracting Symbolic Knowledge from Artificial Neural Networks.* Revised Version of Technical Research Report TR-IAI-93-5, Institut für Informatik III - Universität Bonn.

Towell G., J.W. Shavlik (1993). The Extraction of Refined Rules from Knowledge - Based Neural Networks. *Machine-Learning*, vol.13.

Yoo J.H. (1993). *Symbolic Rule Extraction from Artificial Neural Networks.* PhD thesis, Wayne State University.