LEARNING ALGORITHM FOR RULE-BASED KNOWLEDGE-BASES

Jan Valenta
Doctoral Degree Program (4), FEEC BUT
E-mail: h.valenta@volny.cz

Supervised by: Václav Jirsík
E-mail: jirsik@feec.vutbr.cz

ABSTRACT

This paper describes an algorithm for tuning weights in rule-based knowledge-basis with uncertainty factors. It eliminates thus expensive and time consuming work of knowledge engineer, and eliminates disadvantages of creation knowledge-bases based on neural network. The model of the rule with smooth aggregate functions and the method of learning are presented here.

1. INTRODUCTION

In the last few years the artificial intelligence is fast growing tool for control and diagnostic. They acquit oneself well not only in computer technologies even in the industry field too. A lot of these systems use various knowledge-bases for modeling user behavior or technologic process. The creation such a knowledge-bases is usually most difficult part of developing given system. Plenty of authors use trained neural network (NN) as knowledge base, but is really hard verify such a knowledge-base behavior for non-trained situations. This problem solves rule-based knowledge-basis. Its transparent structure brings easy verification but hard creation and tuning possibilities too. In next few paragraphs is described algorithm which can learn rule-structure from data and thus solve troubles with tuning.

2. EXISTING SOLUTIONS

Some authors try to solve problem with verifying of trained NN, as knowledge-base, using decomposition of NN back to rules. Pioneer of such approach was author Li Ming Fu [1] with his work “Integration of neural heuristic into knowledge-based inference“.

2.1. SEMANTIC CONVERSION INTO NN

The algorithm [1] transforms conjunctive rules in to semantic neural network. It maps the antecedents of the rules to the inputs of the neural net, consequents to output of neural net and hidden rules in to hidden neurons. To keep the semantics one neuron between conditional and consequent part into each rule representing AND function (please refer to Fig. 1) is add. This created structure is in fact a neural network separated into two types of layers: conjunctive – non-differentiable and non-conjunctive – differentiable. The weights in the conjunctive layer are set permanently to 1. The learning algorithm search for the vector of the weights representing uncertainty factors in the structure. This is done by the back propagation algorithm for the non-conjunctive layers and heuristic algorithm hill-climbing for conjunctive layers.

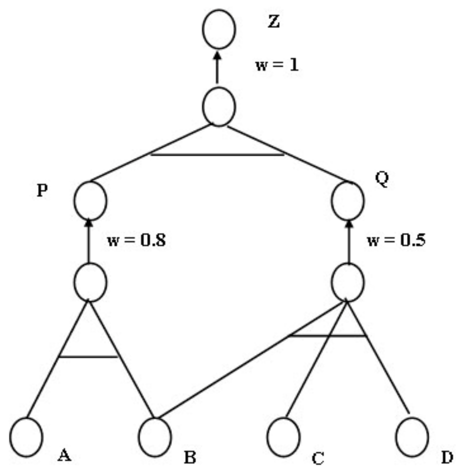

Figure 1: Final form of semantics conversion rules to NN

This simple algorithm allows easy extraction of the rules, but the learning algorithm is not much sophisticated because of the heuristic layers. The number of extracted rules grows with power of neuron inputs.

2.2. KBANN

Another method how to extract rules from trained NN brings couple of authors G. Towell and J. Shavlik.

This algorithm, *K*nowledge-*B*ased *A*rtificial *N*eural *N*etwork, [1], [2], [4] is an algorithm which uses rules for initialization of an NN. The rules are mapped to the NN in the same way as in the previous case (please refer to paragraph 2.1). This method doesn't keep the semantics of the rules – it doesn't add "conjunctives" neurons. The rules structure is converted into NN and the weights are heuristically set to certain value to best approximate AND function. The method adds some new neurons which enables find new knowledge from data. Then the NN is fully interconnected and the weights of these new connections are set to small value.

This common NN can be trained by the BP algorithm. The main disadvantage of this system is loosing semantics of the rules, thereby is very hard to extract the "learned" rules back.

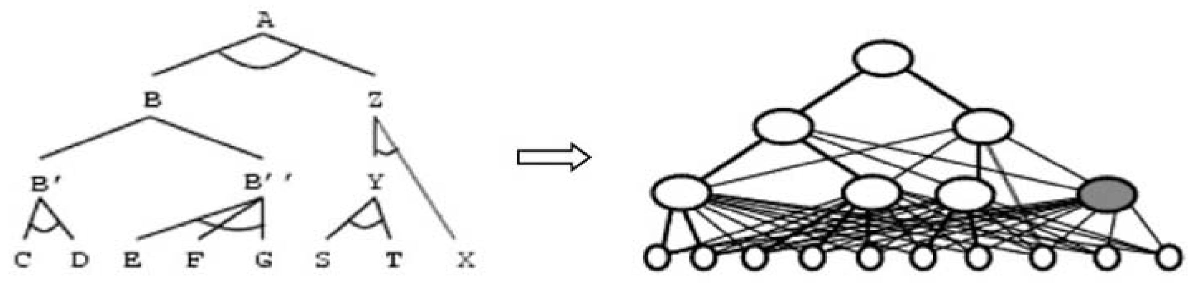

Figure 2: Conversion of the rule structure to NN
(dick lines in NN correspond to the rule structure)

3. DIRECT RULE-STRUCTURE LEARNING METHOD

The systems which use NN as a knowledge-base require complex and time-demanding verification. In this work, is described how to create knowledge-base from the input knowledge acquired from the expert in pure form to keep the knowledge-base structured and verifiable as much as possible.

This work is based on knowledge-base and knowledge structure provided by an expert. The requirements of knowledge-base (KB) are:

- modularity
- transparency
- adaptability
- work with uncertainty

On the other hand, the expert offers knowledge mainly in the form of rules and examples. Comparing the KB requirements and expert knowledge form, the rules are evidently most suitable instrument for knowledge-base creation. It single fills the KB requirements in three of four points. The problem is adaptability here.

We can have a look to rule as a function which combines all its input arguments and the result affects through power of the link (please refer to Figure. 3).

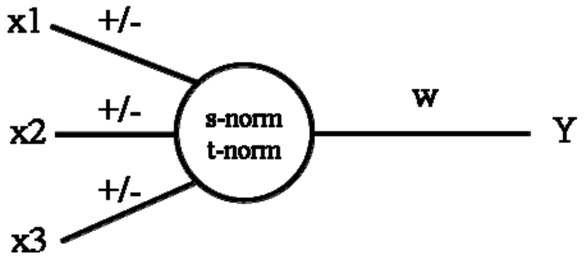

Figure 3: Model of rule

The non-differentiable aggregation functions “AND” realised as minimum of its antecedents and “OR” realised as maximum of its antecedents are substitute by the s-norm for the “AND” function and t-norm for “OR” function. Based on such a model, is possible to create rule structure which work with uncertainty and allow automatic tuning its weights according to patterns acquired from the expert.

3.1. LEARNING ALGORITHM

The algorithm is based on modification of back propagation algorithm [3] to be suitable for rule structure. It adapts weights of each rule according to error on the output error.

The actual quadratic error is

$$e_j = \frac{1}{2}\sum_{p=1}^{M}\left({}_p d_j - {}_p y_j\left(\mathbf{w}_j, {}_p\mathbf{x}_j, \lambda_j\right)\right)^2 = \frac{1}{2}\sum_{p=1}^{M}\left({}_p d_j - \Theta_j\left(\lambda_j^n\left({}_p y_j^{n-1}\right)\right)\right)^2, \tag{1}$$

where M is number of the patterns,

$\boldsymbol{w}$ is vector of the weight of the rule,

λ is function representing positive or negative link to the rule, *Θ* represents s-norm or t-norm function, d_j is relevant pattern,

y_j is output of the structure, $\mathbf{x}$ is vector of the inputs,

j is number of nodes in the given layer of the structure.

The transfer function of the rule is

$$y_j^n = w_j^n \cdot \alpha_j^n = w_j^n \cdot \Theta_j\left(\lambda_j^n\left(y_j^{n-1}\right)\right), \tag{2}$$

where *α* is potential of the rule, i.e. result of *Θ* function.

The gradient of error function depends on the weight *w*

$$\frac{\partial \varepsilon_j^n}{\partial w_j^n} = \frac{1}{2} 2\left(d_j - w_j \Theta\left(\mathbf{x}_j\right)\right)\left(-\Theta\left(\mathbf{x}_j\right)\right) = -\left(d_j - y_j^{n_0}\right) \cdot \alpha_j^{n_0}, \tag{3}$$

where n_0 is output layer

and for the output layer is

$$\frac{\partial \varepsilon_j^{n_0}}{\partial y_j^{n_0}} = d_j^{n_0} - y_j^{n_0}. \tag{4}$$

The weights of the output layer are adapted by equation

$$w_j^{n_0}(t+1) = w_j^{n_0}(t) + \mu_t \alpha_j^{n_0}\left(d_j^{n_0} - y_j^{n_0}\right), \tag{5}$$

the weights in hidden layers by equation

$$w_j^{n-1}(t+1) = w_j^{n-1}(t) + \mu_t \alpha_j^{n-1} \sum_{y_j^{n-1} \to \Theta_j^n} \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot w_j^n \cdot \frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}} \tag{6}$$

where $n \le n_0$.

4. TESTING

This algorithm was implemented into RESLA (*R*ule-based *E*xpert *S*ystem *L*earning *A*lgorithm) expert system for verifying its work on real tasks.

The first task on which the algorithm was tested where, logic functions AND, OR and XOR. The structure of rules is as given:

$A \wedge \neg B \to C\ [0.5]$

$\neg A \wedge B \to D\ [0.5]$

$A \to E\ [0.5]$

$B \to F\ [0.5]$

$C \vee D \to Y_{XOR}\ [0.5]$

$E \wedge F \to Y_{AND}\ [0.5]$, where *A* and *B* are inputs, other characters represent hidden nodes.

$E \vee F \to Y_{OR}\ [0.5]$

The value in the brackets is initial uncertainty of the rule. The pattern for this structure was set to (A,B $\to$ Y_{AND}, Y_{OR}, Y_{XOR}):

1 , 1 $\to$ 1, 1, 0

0 , 1 $\to$ 0, 1, 1

The require values of weights on all rules (by the initialization set to 0.5) is 1.0. The algorithm tune weights in 31 steps with absolute average error less than 0.1 % for training patterns. For the testing pattern (input values between 0 and 1, desired values calculated using this structure with weights set to 1.0) is error les than 0.1 % too. Adding next pattern to train set

1 , 0 → 0, 1, 1 , the number of cycles descend to 23 with the same accuracy.

The algorithm was tested on knowledge-base for identification part of heart malfunction (brady-arrhythmia) as a real task. The KB consist of 7 inputs (questions), e.g. length of PR interval, width of QRS, the shape of QRS, which lead is biggest R-wave, etc., 21 hidden nodes realizing conjunctive or disjunctive rule and 17 hypothesis, e.g. AV block 1st level, bifascikular blockade, IRBB, LBBB, left-front hemi-block etc.

The patterns are set according to the expert knowledge. The algorithm tunes this KB in 39 steps with absolute average error 1 % for training patterns.

5. CONCLUSION

The algorithm presented in this paper shows possibilities how to create rule knowledge-bases for expert systems directly from the data. It eliminates thus expensive and time consuming work of knowledge engineer, and eliminates disadvantages of creation knowledge-bases based on neural network. The rule structure must be composite only of smooth functions therefore the AND and the OR functions of the rules are substituted by s-norm and t-norm. Presented learning method is faster than semantic conversion of neural network and KBANN because it doesn't need extract the rules back from the network. The algorithm was successfully tested on several tasks (logical and real) with the result better than 1 % for all knowledge-basis.

ACKNOWLEDGEMENT

This work has been supported in part by Ministry of Education, Youth and Sports of the Czech Republic (Research Intent MSM0021630529 Intelligent systems in automation, and by the Brno University of Technology. Without kind support of the above-mentioned agencies and institutions, the presented research and development would not be possible.

REFERENCES

[1] BUZING, P.: "Hybrid systems: Two examples of the combination of rule-based systems and neural nets", 2001, available at http://pds.twi.tudelft.nl/~buzing/Articles/hybrid.doc, Sep. 2007.

[2] Bose R., Nagaraja G.: "Performance Studies on KBANN", Proceedings of the Fourth International Conference on Hybrid Intelligent, IEEE Computer Society, 2004.

[3] KASABOV N.: "Foundations of neural networks, fuzzy systems, and knowledge engineering", MIT Press, London, 1996.

[4] TOWELL G., SHAVLIK J.: "Knowledge-based artificial neural networks", Artificial intelligence, Vo. 70, Elsevier Science Publishers Ltd., Essex, 1994.