Neural optimization of linguistic variables and membership functions

Włodzisław Duch, Rafał Adamczak and Krzysztof Grąbczewski
Department of Computer Methods, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland.
E-mail: duch,raad,kgrabcze@phys.uni.torun.pl

Abstract— **Algorithms for extraction of logical rules from data that contains real-valued components require determination of linguistic variables or membership functions. Context-dependent membership functions for crisp and fuzzy linguistic variables are introduced and methods of their determination described. Methodology of extraction, optimization and application of sets of logical rules is described. Neural networks are used for initial determination of linguistic variables and rule extraction, followed by minimization procedures for optimization of the sets of rules. Gaussian uncertainties of measurements are assumed during application of crisp logical rules, leading to "soft trapezoidal" membership functions and allowing to optimize the linguistic variables using gradient procedures. Applications to a number of benchmark and real life problems yield very good results.**

Keywords— **Logical rules, linguistic variables, neural networks, data mining, medical diagnosis**

I. Introduction.

A GOOD strategy in data mining and classification tasks is to use the simplest description of the data that is possible without compromising accuracy. It is advantageous to extract crisp logical rules first, use fuzzy rules if crisp rules are not sufficient and only if the number of logical rules required for high accuracy of classification is too large use other, more sophisticated tools. In many applications simple crisp logical rules proved to be more accurate and were able to generalize better than many machine and neural learning algorithms [1]. In other applications fuzzy rules proved to be more accurate [2]. There is no reason why a simple model based on logical rules should always work, but in cases when it does it is certainly worth using. Extraction of logical rules from data may be done using statistical, pattern recognition and machine learning methods, as well as neural network methods [3]. Recently we have presented a complete methodology for extraction, optimization and application of logical rules [2], [4]. The last two steps are largely neglected in the literature, with current emphasis being still on the extraction methods.

Logical rules require linguistic variables. Selection of linguistic variables for symbolic attributes is simple but for real-valued attributes may be difficult. In such cases rule extraction process should be done in an iterative way: initial linguistic variables are defined, rules extracted, linguistic variables optimized and the extraction process repeated until the whole procedure converges to a stable set of rules and linguistic variables.

In the next section different types of linguistic variables are described and methods of their determination presented. The third section describes briefly our methodology of logical rule extraction, stressing the issues of linguistic variable optimization. The forth section discusses determination of membership functions in case of uncertainties in input values. Analytical formulas reproducing Monte Carlo classification probabilities are given for this case. Results on benchmark and real-world datasets are presented in the fourth section and in the last section conclusions are given.

II. Context-dependent linguistic variables.

LINGUISTIC variables may be introduced in several ways. A symbolic attribute *color* may take values *green, red, blue* and appear in a rule as logical condition, for example *color=red*. An alternative way is to use a predicate function *color(x)*. Depending on the type of variable *x* predicate function may have different interpretation. For example, if *x* is the wavelength of light and $x \in [600\ nm, 700\ nm]$ then *color(x)* is *red*, i.e. logical condition *color(x)=red* is true. One may also introduce predicates for each color defined by logical functions *color-green(x), color-red(x), color-blue(x)*. Such logical predicate functions are linguistic variables, mapping symbolic or real values of *x* into binary 0, 1 or *false, true*.

If the input $x \in X$ is given as a real number or a large number of integer or symbolic values linguistic variables are created dividing the data range X into distinct (for crisp logic) sets X_i and introducing variables $s_i(x)$ =F unless $x \in X_i$, when $s_i(x)$ =T. For $X \subseteq R$ sets X_i are usually intervals and linguistic variables are binary functions mapping *x* into 0 or 1. A typical linguistic variables associated with the tire pressure attribute will be *low* if $x < 1.7$, *normal* if $1.7 \leq x \leq 2.2$ and *high* if $x \geq 2.2$. A rule may then have conditions of the form *high(x)*, which is usually written as *x=high*, meaning that $x \geq 2.2$.

In many applications a common set of linguistic variables is assumed for all rules. This is certainly a great simplification since linguistic variables are always **context dependent**. Tire pressures for bicycles are different than for cars or trucks. Thus instead of using a fixed number of linguistic variables partitioning the whole attribute range one should rather use rule-dependent linguistic variables, optimized for each rule.

Introducing *color-red(x)* predicate that has values in the $[0, 1]$ range, instead of the binary 0, 1 values, one may interpret it as estimation of similarity of color that *x* has to the typical red color. Using such predicate functions as logical conditions is equivalent to some form of fuzzy logic, depending on the way logical functions are mapped into arithmetic functions [5]. Thus predicate functions play the role of membership functions: binary valued are used in crisp logic and real valued in fuzzy logic (for multistep values multivalued logic conditions are defined). For $X \subseteq R$ crisp membership functions are rectangular while fuzzy membership functions have triangular, trapezoid, Gaussian or other shapes useful for evaluation of similarities. Quite frequently a number of linguistic variables is defined for each attribute, for example 3 triangular membership functions per attribute, $\mu_1(x_i), \mu_2(x_i), \mu_3(x_i)$, and rules for combinations

IF $(\mu_{k_1}(x_1) \wedge \mu_{k_2}(x_2) \ldots \wedge \mu_{k_N}(x_N))$

are sought [5], with $k_i = 1,2,3$. Unfortunately the number of combinations grows exponentially with the number of attributes (here 3^N) and the method works only for 2 or 3 dimensions. Good results with such linguistic variables may be obtained only by chance. Fuzzy rules with fixed membership functions give decision borders that are not much more flexible than those of crisp rules. In both crisp and fuzzy case linguistic variables should be **context dependent**, i.e. different in each rule. For example if x_1=*broad* for $1 \leq x_1 \leq 4$, x_1=*average* for $2 \leq x_1 \leq 3$, and x_2=*small* for $1 \leq x_2 \leq 2$, x_2=*large* for $3 \leq x_2 \leq 4$ then two simple rules:

IF(x_1=*broad*$\wedge x_2$=*small*) THEN C=*great*
IF(x_1=*average*$\wedge x_2$=*large*) THEN C=*great*
ELSE C=*so-so*

would be more complex if written using linguistic variables that partition x_1 into distinct or just partially overlapping subsets. In this case one may say that in the context of x_2=*large* linguistic variable x_1 =*average*, rather than *broad*, should be used.

The simplest way to select initial linguistic variables is to analyze histograms, displaying data for all classes for each attribute. For the attribute x_i good intervals useful to distinguish two classes are found only if the histograms for these classes do not overlap. Unfortunately frequently this is not the case. Therefore we have developed several neural methods for determination of linguistic variables.

Feature Space Mapping (FSM) is a constructive neural network [6] that estimates probability density of vectors in each class. Nodes of this network use localized, separable transfer functions, providing good linguistic variables. Crisp decision regions are obtained by using rectangular transfer functions. The network is initialized using a clusterization method [7], and adapted to the incoming input data by moving, decreasing and increasing the nodes, or by adding new nodes if it is necessary. The FSM network may use any separable transfer functions, including triangular, trapezoidal, Gaussian, or the biradial combinations $\prod_i(\sigma(x_i - b_i) - \sigma(x_i + b'_i))$ of sigmoidal functions [8] with soft trapezoidal shapes. If gain of sigmoidal functions $\sigma(x)$ is slowly increased during learning rectangular functions are smoothly recovered.

Another approach is based on general separability criterion introduced recently [9]. The best split value is the one which separates the maximal number of pairs of vectors from different classes and among all the split values which satisfy this condition – the one which separates the smallest number of pairs of vectors belonging to the same class. The criterion is useful for creation of decision trees as well as determination of linguistic variables.

Good linguistic variables are also obtained in an adaptive way using multilayer perceptron network with some constraints. We have proposed special neural units, called *L*-units (linguistic units), that automatically analyze continuos inputs and produce linguistic variables. The basic scheme of such unit is shown in Figure 1. An input x_i is connected via W_1, W_2 weights to two neurons, each with it own separate bias, b_i and b'_i. All transfer functions are sigmoids that, at the end of training, become very steep, although at the beginning they may be quite smooth, allowing for fuzzy approximation of features. The two hidden neurons of the *L*-unit are connected to its output neuron using weights S_1, S_2. Experiments showed that learning is faster if connections from the two hidden L-unit neurons to other hidden neurons of the MLP network are added. All weights have values constrained at the end of the training to $0, \pm 1$. The network with L-units and hidden units (called R-units, since they will provide logical rules) is an MLP network with specific architecture. Since *L*-units have only one input, one output and four constrained weights as parameters functions realized by these units belong to one of the four types shown in the limit of large gain in Figure 1. The first of these functions is obtained as a differences of two sigmoids and represents a typical linguistic variable s_k equivalent to $x_i \in [b_i, b'_i]$, the second denotes negation $\neg s_k$ while the other two, with only one non-zero weight, correspond to $x_i \geq b$ or $x_i \leq b$. The borders b_i and b'_i defining linguistic variables and the four constrained weights are treated as adaptive parameters of our network.

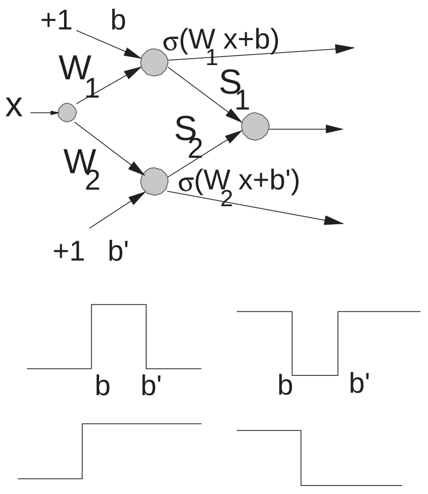

Fig. 1. Construction of a linguistic unit converting continuous inputs to linguistic variables.

The threshold of the output unit is kept fixed at one. Input weights W_1, W_2, and the weights S_1, S_2, each taking values constrained to $0, \pm 1$, may take at most 27 values, giving for each combination an L-unit transfer function. Most of these output functions are identically zero. We have found that training L-units separately from R-units is easier, i.e. when the L-units weights are trained (optimization of linguistic variables) R-unit weights are frozen and vice versa. It frequently happens that the output L-unit neuron has both weights $S_1, S_2 = 0$ and should be deleted, because open intervals realized by the hidden L-unit nodes are sufficient.

III. Neural rule extraction methodology

In our C-MLP2LN algorithm small MLP network is initially created, usually with one hidden neuron and several L-units. Training is done using a constructive MLP algorithm (C-

MLP), increasing the complexity to account for the incoming data. To transform an MLP into a network performing logical operations (Logical Network, LN) new neurons are trained minimizing the standard mean error function plus two penalty terms, one $\sum_{i,j} W_{ij}^2$ scaled by the λ_1 hyperparameter, encouraging weight decay and leading to skeletonization of the network and elimination of irrelevant features, and the second, $\sum_{i,j} W_{ij}^2 (W_{ij}-1)^2 (W_{ij}+1)^2$, scaled by λ_2, forcing the remaining weights to approach ± 1 or 0, facilitating easy logical interpretation of the network function: 0 = irrelevant input, $+1$ = positive and -1 = negative evidence. The slope of sigmoidal functions in L-units increases during training. The hyperparameters determine the simplicity/accuracy tradeoff of the generated network and extracted rules. If very simple networks (logical rules) are desired, giving only rough description of the data, λ_1 is increased until error grows sharply.

After training the first neuron is finished (in K-class problem $K-1$ neurons are trained at the same time) connections with zero weights are deleted and the skeleton network is kept frozen. The second neuron is added, connected to a new set of L-units. This assures that context-dependent linguistic variables are created and used in logical rules extracted from the network after training. During the learning process the network expands when neurons are added and shrinks when connections are deleted. This procedure is repeated until most of the data is correctly classified or the number of new neurons starts to grow rapidly, indicating overfitting and noise in the data.

An iterative optimization process is used: neural networks are initialized randomly, trained, linguistic inputs are analyzed, logical rules extracted, intervals defining linguistic variables optimized using sets of rules (see below), and the whole process repeated until convergence is achieved. Usually two or three iterations are sufficient.

Increasing the slopes of sigmoidal functions transforms complex decision regions into simpler, hypercuboidal decision regions. **Rules R_k equivalent to logical functions** performed by MLP networks are obtained in the form of logical conditions by considering contributions of inputs for each linguistic variable s_i. A combination of linguistic variables activating the hidden neuron above the threshold is a logical rule of the form: $R = (s_1 \wedge \neg s_2 \wedge ... \wedge s_k)$. After analysis of all ± 1 weights a set of rules $R_1 \vee R_2 ... \vee R_n$ is found for each output class. Rules obtained by the C-MLP2LN algorithm are ordered, starting with rules that cover many cases and ending with rules that cover only a few cases.

Simplification of rules: some rules obtained from analysis of the network may involve spurious conditions, more specific rules may be contained in general rules or logical expressions may be simplified if written in another form. We use a Prolog program for the simplification step.

Optimization of rules: optimal intervals for linguistic variables are found by maximization of a predictive power of the rule-based classifier. Let $P(C_i, C_j|M)$ be the confusion matrix, i.e. the number of instances in which class C_j is predicted when the true class was C_i, given the model M. Then for n samples $\mathbf{p}(C_i, C_j|M) = P(C_i, C_j|M)/n$ is the probability of (mis)classification. The best parameters of the model M are selected by maximizing the "predictive power" of rules $\max_M [\mathrm{Tr}\, P(C_i, C_j|M)]$ over all parameters M, or by minimizing the number of wrong predictions (possibly with some risk matrix $\mathbf{R}(C_i, C_j)$), $\min_M \left[\sum_{i\neq j} \mathbf{R}(C_i, C_j) P(C_i, C_j|M)\right]$. Weighted combination of these two terms is used:

$$E(M) = \gamma \sum_{i\neq j} P(C_i, C_j|M) - \mathrm{Tr}\, P(C_i, C_j|M) \geq -n \qquad (1)$$

minimized over parameters M without constraints. If γ is large the number of errors after minimization may become zero but some instances may be rejected (i.e. rules will not cover the whole input space). An alternative to minimization of the whole set of rules simultaneously is to define a cost function for each rule separately and minimize it over parameters M used only in the single rule R. We have used two global minimization techniques for this optimization [10] but the method described below allows to use more efficient gradient procedures.

Neural systems have good generalization properties because they are wide margin classifiers. Their decision borders are obtained from the mean square error optimization of smooth function that extends over larger neighborhood contributing to the error. This allows for three important improvements: the use of inexpensive gradient method instead of global minimization, more robust rules with wider classification margins, and probabilistic estimation of confidence. Input values result usually from observations which are not quite accurate, therefore instead of the attribute value x a Gaussian distribution $G_x = G(y; x, s_x)$ centered around x with dispersion s_x should be given. This distribution may be treated as a membership function of a fuzzy number G_x. A Monte Carlo procedure may be performed sampling vectors from Gaussian distributions defined for all attributes to compute probabilities $p(C_i|X)$. Analytical evaluation is based on the cumulative distribution function:

$$\rho(a-x) = \int_{-\infty}^{a} G(y; x, s_x) dy = \qquad (2)$$
$$\frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{a-x}{s_x\sqrt{2}}\right)\right] \approx \sigma(\beta(a-x))$$

where erf is the error function and $\beta = 2.4/\sqrt{2} s_x$ makes the erf function similar to the standard unipolar sigmoidal function with the accuracy better than 2%. A rule $R_a(x)$ with single crisp condition $x \geq a$ is fulfilled by a Gaussian number G_x with probability:

$$p(R_a(G_x) = T) = \int_{a}^{+\infty} G(y; x, s_x) dy \approx \sigma(\beta(x-a)) \qquad (3)$$

Taking instead of the erf function a sigmoidal function changes assumption about the error distribution of x from Gaussian to $\sigma(x)(1-\sigma(x))$, approximating Gaussian with $s^2 = 1.7$ within 3.5%. If the rule involves closed interval $[a, b], a \leq b$ the probability that it is fulfilled by a sample from the Gaussian distribution representing the data is:

$$p(R_{a,b}(G_x) = T) \approx \sigma(\beta(x-a)) - \sigma(\beta(x-b)) \qquad (4)$$

Thus the probability that a given condition is fulfilled is proportional to the value of soft trapezoid function realized by L-unit. Crisp logical rules with assumption that data has been measured with finite precision lead to soft L-functions that allow to compute classification probabilities that are no longer binary. In this way we may either fuzzify the crisp logical rules or obtain fuzzy rules directly from neural networks.

It is easy to calculate probabilities for single rule conditions of the form $x < a$, $x > a$ or $x \in (a,b)$:

$$\begin{aligned} P(x<a) &= \int_{-\infty}^{a} G(y;x,s_x)dy = \frac{1}{2}\left[1+\operatorname{erf}\left(\frac{a-x}{s_x\sqrt{2}}\right)\right] \\ P(x>a) &= \int_{a}^{+\infty} G(y;x,s_x)dy = \frac{1}{2}\left[1-\operatorname{erf}\left(\frac{a-x}{s_x\sqrt{2}}\right)\right] \\ P(x\in(a,b)) &= \frac{1}{2}\left[\operatorname{erf}\left(\frac{b-x}{s_x\sqrt{2}}\right)-\operatorname{erf}\left(\frac{a-x}{s_x\sqrt{2}}\right)\right] \end{aligned}$$

Notice that this interpretation does not differentiate inequalities $\leq$ and $<$ so to obtain reasonable probabilities rules with borders such that $\leq$ may be replaced by $<$ without loss of accuracy are required.

The probability that x belongs to a rule $R = r_1 \wedge \ldots \wedge r_N$ may be defined as the product of the probabilities of $x \in r_i$ for $i = 1, ...N$. Such definition assumes that all the attributes which occur in rule R are mutually independent, which is usually not the case. However, if the rule generator produces as simple rules as possible there should be no pairs of strongly dependent attributes in a single rule. Therefore the product should be very close to real probability. Obviously the rule may not contain more than one premise per one attribute, but it is easy to convert the rules appropriately if they do not satisfy this condition.

Another problem occurs when probability of x belonging to a class described by more than one rule is estimated. Rules usually overlap because they use only a subset of all attributes and their conditions do not exclude each other. Summing and normalizing probabilities obtained for different classes may give results quite different from real Monte Carlo probabilities. To avoid this probabilities are calculated as:

$$P(x \in C) = \sum_{R \in 2^{R_C}} (-1)^{|R|+1} P(x \in \bigcap R) \tag{5}$$

where R_C is the set of the classification rules for class C and $|R|$ is the number of elements in R.

Instead of the number of misclassifications the error function may include a sum over all probabilities:

$$E(M,s_x) = \frac{1}{2}\sum_X \sum_i \left(p(C_i|X;M) - \delta(C(X),C_i)\right)^2 \tag{6}$$

where M includes intervals defining linguistic variables and s_x are Gaussian uncertainties of inputs. Confusion matrix computed using probabilities instead of the number of errors allows for optimization of Eq. (1) using gradient-based methods. This minimization may be performed directly or may be presented as a neural network problem with special network architecture. Uncertainties s_x of the values of features are additional adaptive parameters that may be optimized. We have used so far a very simple optimization with all s_x taken as a percentage of the range of feature x to perform one dimensional minimization of the error function independently of other steps. This approach to soft optimization may be used with any crisp logical rules to overcome the brittleness problem and to obtain robust wide margin rule-based classifiers.

IV. Illustrative Applications

We have analyzed a large number of datasets comparing our results with the results obtained by other methods whenever possible. Many results are collected at:
http://www.phys.uni.torun.pl/kmk/projects/rules.html
Web page. Here only a few new results obtained due to the improvements in optimization of linguistic variables are shown. In most cases for the two-class problems only one L-unit was sufficient, with non-zero weights connected to a single hidden L-unit neuron. Therefore the final rules are very simple.

A. Wisconsin breast cancer data.

We have already analyzed the Wisconsin cancer dataset before giving detailed comparison with other results [1]. The data has 699 instances, with 458 benign (65.5%) and 241 (34.5%) malignant cases. The values of the 9 attributes are quantized in the range 1-10. The simplest rules for malignant class obtained from optimization offer the overall accuracy 94.9%

$$f_2 \geq 7 \vee f_7 \geq 6 \tag{7}$$

Using L-units 4 more accurate rules for the malignant class are created (their reliability is in parenthesis):

R_1: $f_4 < 3 \wedge f_5 < 4 \wedge f_7 < 6 \wedge f10 = 1$ (99.5)%
R_2: $f_2 < 7 \wedge f_5 < 4 \wedge f_7 < 6 \wedge f10 = 1$ (99.8)%
R_3: $f_2 < 7 \wedge f_4 < 3 \wedge f_7 < 6 \wedge f10 = 1$ (99.4)%
R_4: $f_2 < 7 \wedge f_4 < 3 \wedge f_5 < 4 \wedge f_7 < 6$ (99.4)%

that (includeing ELSE condition) give 97.7% overall accuracy. The confusion matrix (benign, malignant) is $\begin{pmatrix} 447 & 5 \\ 11 & 236 \end{pmatrix}$ (only 5 malignant cases are misclassified as benign). More complex rules with 99.0% overall accuracy have also been found [1], reducing the number of benign misclassification to 1 and leaving 6 misclassified malignant cases. Fuzzified rules predict with almost 100% confidence that these vectors belong to the wrong class, indicating that the data is really noisy.

B. Diabetes.

This is the “Pima Indian diabetes” dataset from UCI repository [11]. Previously we were not successful in extracting rules for this dataset because histograms are do not provide a useful starting point here. All patients were females at least 21 years old, of Pima Indian heritage. The data contains 2 classes, 8 attributes, 768 instances, 500 (65.1%) healthy and 268 (34.9%) diabetes cases. This dataset was used in the Statlog project [14], with the best 10-fold crossvalidation accuracy around 77.7% obtained by logistic discriminant analysis. One simple rule for the healthy cases achieving 75% accuracy is:

$$f_2 \leq 151 \wedge f_6 \leq 47 \tag{8}$$

where f_2 is the “plasma glucose concentration” and f_6 the body mass index (weight in kg/(height in m)^2). The confusion matrix (healthy, diabetes) is: $\begin{pmatrix} 467 & 159 \\ 33 & 109 \end{pmatrix}$.

C. Psychometric data

The methods described here were tried in a real-world data mining task, providing logical description of the psychometric data collected in the Academic Psychological Clinic of Nicholas Copernicus University and in several psychiatric hospitals around Poland. Minnesota Multiphasic Personality Inventory (MMPI) test was used. The final two databases analyzed had over 1000 cases each. Standard evaluation of such data is based on aggregation of answers into 14 coefficients, called "psychometric scales". These coefficients are often displayed as a histogram (called "a psychogram") allowing skilled psychologists to diagnose specific problems, such as neurosis, drug addiction or criminal tendencies. Our goal was to provide an automatic psychological diagnosis.

Rule based system is most desirable because a detailed interpretation, including description of personality type, may be assigned to each diagnosis. We have worked with datasets containing up to 34 classes (normal, neurotic, drug addicts, schizophrenic, psychopaths, organic problems, malingerers etc.) determined by expert psychologists. Our initial logical rules achieved about 93% accuracy on the whole set, increasing to over 95%-97% after some optimization and fuzzification. About 50-60 rules were generated for each datasets (man, woman, mixed cases). On average 2.5 logical rules per class were derived, involving between 2 and 9 attributes. For most classes there were only a few errors and it is quite probable that they are due to the psychologists interpreting the psychogram data. Two classes, organic problems and schizophrenia, are difficult since their symptoms are easily confused with symptoms belonging to other classes.

A typical rule has conditions referring to the values of scales as $Hy(X) \in [72,88]$ (Hypochondria scale) or $Ps \in [74,80]$ (psychostenia scale) and therefore may easily be interpreted in a verbal way. Each rule has detailed interpretation associated with it by psychologists. These rules are used with an assumption about finite accuracy of the measurement, in each of the scales corresponding to a Gaussian dispersion of about 5 units and leading to some fuzzification of membership functions. This leads to additional adjectives in verbal interpretation, like "strong tendencies", or "typical". An expert system using these rules should be evaluated by clinical psychologist in the near future.

V. Conclusions

In this paper neural methods of determination of linguistic variables have been presented. The problem is not separable from the rule extraction itself. An iterative algorithm is used, improving in turns linguistic variables and then rules using these variables. We have stressed the importance of context-dependent linguistic variables since an unwarranted assumption that the whole range of attribute values should be partitioned into intervals corresponding to linguistic variables is frequently used. Complete methodology, based on neural networks, for construction of logical rules has been briefly described. Optimization and application of rules is often neglected in other approaches. We have derived analytical formulas allowing to find the best fuzzification of the crisp membership functions that will give the same probabilities as the Monte Carlo procedure performed for input vectors distributed around measured values in a Gaussian way.

Using the early version of this methodology simplest logical description for several benchmark problems was obtained [1]. Now we were also able to find simple rules for datasets that could not be analyzed earlier (see also [15]). It is not quite clear why, for some medical datasets, logical rules work better than any other method [2]. One reason for such performance of rule-based systems is due to the good control of the complexity of data representation. Another possible explanation for the medical data is that the classes labeled "sick" or "healthy" have really fuzzy character. If the doctors are forced to make yes-no diagnosis they may fit the results of tests to specific intervals, implicitly using crisp logical rules.

We are sure that in all cases, independently of the final classifier used, it is advantageous to extract crisp logical rules. First, in our tests logical rules proved to be highly accurate; second, they are easily understandable by experts in a given domain; third, they may expose problems with the data itself. However, if the number of rules is too high or the accuracy of classification is too low, other methods should be attempted.

Acknowledgments

Support by the KBN, grant 8 T11F 014 14, is gratefully acknowledged.

References

[1] W. Duch, R. Adamczak, K. Grąbczewski, G. Żal, Hybrid neural-global minimization logical rule extraction method for medical diagnosis support, Intelligent Information Systems VII, Malbork, Poland, 15-19.06.1998, pp. 85-94

[2] W. Duch, R. Adamczak, K. Grąbczewski, G. Żal, Y. Hayashi, Fuzzy and crisp logical rule extraction methods in application to medical data. Computational Intelligence and Applications. Springer Studies in Fuzziness and Soft Computing, Vol. 23 (ed. P.S. Szczepaniak), in print (1999)

[3] R. Andrews, J. Diederich, A.B. Tickle, "A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks", Knowledge-Based Systems 8 (1995) 373–389

[4] W. Duch, R. Adamczak, K. Grąbczewski, G. Żal, Hybrid neural-global minimization method of logical rule extraction, Int. Journal of Advanced Computational Intelligence (in print)

[5] B. Kosko, Neural Networks and Fuzzy Systems. Prentice Hall 1992

[6] W. Duch, G.H.F. Diercksen, Feature Space Mapping as a universal adaptive system, Computer Physics Communication 87 (1995) 341–371

[7] W. Duch, R. Adamczak, N. Jankowski, Initialization of adaptive parameters in density networks, 3-rd Conf. on Neural Networks, Kule, Oct. 1997, pp. 99-104

[8] W. Duch, R. Adamczak, N. Jankowski, New developments in the Feature Space Mapping model, 3rd Conf. on Neural Networks, Kule, Poland, Oct. 1997, pp. 65-70

[9] K. Grąbczewski, W. Duch, *A general purpose separability criterion for classification systems.* Fourth Conference on Neural Networks and Their Applications, Zakopane, May 1999 (in print)

[10] W. Duch, R. Adamczak and K. Grąbczewski, Extraction of logical rules from backpropagation networks. Neural Processing Letters 7, 1-9 (1998)

[11] C.J. Mertz, P.M. Murphy, UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-data-bases.

[12] S.M. Weiss, I. Kapouleas, An empirical comparison of pattern recognition, neural nets and machine learning classification methods. In: J.W. Shavlik and T.G. Dietterich, *Readings in Machine Learning*, Morgan Kauffman Publ, CA 1990

[13] W. Duch, R. Adamczak and K. Grąbczewski, Constraint MLP and density estimation for extraction of crisp logical rules from data. ICONIP'97, New Zealand, Nov.1997, pp. 831-834

[14] D. Michie, D.J. Spiegelhalter and C.C. Taylor, Machine learning, neural and statistical classification. Elis Horwood, London 1994

[15] W. Duch, R. Adamczak, K. Grąbczewski, Optimization of Logical Rules Derived by Neural Procedures, International Joint Conference on Neural Networks, Washington, 10-16 June 1999 (in print)