Optimization of Logical Rules Derived by Neural Procedures

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.

Neural networks are used for initial determination of linguistic variables and extraction of logical rules from data. Hierarchical sets of rules for different accuracy/rejection tradeoffs are obtained. Sets of logical rules are optimized by maximization of their predictive power. To avoid global minimization methods for crisp logical rules Gaussian uncertainties of inputs are assumed. Analytical formulas reproducing Monte Carlo results for such inputs are derived, leading to a "soft trapezoidal" membership functions instead of rectangular functions used for crisp logical rules. Such approach increases accuracy, gives probabilities of classification instead of the yes/no answers, and allows to optimize sets of rules using gradient procedures. A few illustrative applications to benchmark and real life problems show the effectiveness of this approach.

Introduction.

IN DATA mining and classification problems one should use the simplest description of the data that is possible without compromising accuracy. Description of the data using crisp logical rules should be attempted first, and only if it is not satisfactory – because of the low classification accuracy or because the number of logical rules becomes too large – fuzzy rules may be used, followed by other, more sophisticated tools, such as neural networks. 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. In many medical applications simple crisp logical rules proved to be more accurate and were able to generalize better than many machine and neural learning algorithms [1], [2]. As we will show in this paper crisp rules may be converted to a specific form of fuzzy rules and optimized using gradient procedure, providing higher accuracy without significant increase of the complexity of the classification system.

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. Previously we have used global minimization methods for optimization of linguistic variables for real-valued attributes. Although such method work well they are computationally quite expensive. Estimation of reliability of sets of logical rules has been done by exploiting accuracy/rejection tradeoff [2]. Hierarchical sets of rules are created, starting from rules that are very reliable but reject many cases (leaving them as unclassified) to rules that classify all data but are less reliable. Extraction of sets of rules may be tedious. For a given case crisp logical rules always assign only a single class (with probability for each class equal to 0 or 1), even in cases when other classifiers would report similar probability for two or more classes.

All these drawbacks are overcomed in this paper. Assuming that the input vectors are obtained from measurements performed with finite precision one can calculate probabilities for rule-based or any other classifier by Monte Carlo procedure, repeating calculations for input vectors sampled around the measured values. Assuming Gaussian uncertainties allows to derive analytical formulas for classification probabilities. Minimization of error function for soft probabilities may be done using efficient gradient procedures allowing to optimize linguistic variables for very large sets of rules. The ease of interpretation – the main reason to use crisp logical rules – is not sacrificed since the rules are still crisp, only the inputs are Gaussian fuzzy vectors. Such approach is equivalent to the use of fuzzy rules with "soft trapezoid" membership functions applied to crisp input vectors. One additional parameter, percent of input uncertainty for each (standardized) attribute, frequently brings significant improvements and may be treated as an adaptive parameter, obtained from minimization of error. Of course one may use more parameters: a separate uncertainty for each attribute, different uncertainties in different regions of the inputs space (covered by different logical rules).

In the next section our methodology of logical rule extraction is briefly described. In the third section optimization methods of the sets of rules are discussed. Results on benchmark

and real-world datasets are presented in the fourth section and in the last section conclusions are given.

Neural rule extraction methodology

NEURAL methodology of crisp logical rule extraction developed by our group has been described in a series of papers [1], [2], [4], [10], [13], therefore only a very brief summary is given here.

Selection of linguistic variables. Linguistic variables used by us are **context dependent**, i.e. they may be different in each rule (more on the linguistic variables [14]. For real-valued attributes intervals defining linguistic variables used in logical rules are needed. Initialization of these intervals is done by: analysis of histograms (works only in simple cases), using a general separability criterion [9] (similar to criteria used in decision-trees), using Feature Space Mapping (FSM) constructive neural network [6], or using special "linguistic units" (L-units) in an MLP (multilayer perceptron) network [14]. The FSM neural network uses arbitrary separable transfer functions, including triangular, trapezoidal, Gaussian, or the bicentral combinations $\prod_i(\sigma(x_i-b_i)-\sigma(x_i+b_i'))$ of sigmoidal functions [8] with soft trapezoidal shapes. If the slope of sigmoidal functions $\sigma(x)$ is slowly increased during learning rectangular functions are smoothly recovered. After training nodes of FSM network are analyzed providing good intervals for logical variables.

Linguistic neural units (L-units) automatically analyze continuos inputs and produce linguistic variables [14]. The basic idea is based on using a window-type functions provided by two neurons, each with it own separate bias, b_i and b_i', defining the linguistic variable intervals. Differences of two sigmoids represent a typical linguistic variable s_k equivalent to $x_i \in [b_i, b_i']$, its negation $\neg s_k$ while a single sigmoid may realize $x_i \geq b$ or $x_i \leq b$ variables. The borders b_i and b_i' defining linguistic variables are adaptive parameters of the MLP network. 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. All weights have values constrained at the end of the training to $0, \pm 1$. The network with L-units and a second layer of hidden units (called R-units, since they will provide logical rules) is an MLP network with specific architecture.

Neural architectures. In the 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 by minimization of the standard mean error function plus penalty terms forcing the weights to approach ± 1 or 0, i.e. encouraging weight decay, leading to skeletonization of the network, elimination of irrelevant features and facilitating easy logical interpretation of the network function: 0 = irrelevant input, $+1$ = positive and -1 = negative evidence. The regularization hyperparameters determine the simplicity/accuracy tradeoff of the generated network and extracted rules.

After training the first neuron 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. 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: $R = (s_1 \wedge \neg s_2 \wedge \ldots \wedge s_k)$. After analysis of all ± 1 weights a set of rules $R_1 \vee R_2 \ldots \vee R_n$ 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.

Although C-MLP2LN algorithm works very well (especially with optimization of final rules described below) in complex cases FSM network with rectangular functions (or soft rectangular functions that are changed into rectangular during training) is frequently simpler to use. FSM uses good clusterization procedures (based on dendrograms or decision trees) for initialization, frequently obtaining quite good results without any training [7], [8]. After training nodes that cover only a few training vectors are removed and nodes that cover many training vectors are optimized. The node covering the largest number of vectors, assigned to class C_i, is selected (this node corresponds to the most general logical rule); choosing feature $k = 1 \ldots n$ activity of this node is checked using the remaining $n-1$ inputs (this is possible because transfer functions used by FSM network are separable). The interval $[b_k, b_k']$ for the selected node is adjusted to cover all C_i class vectors that activate it. The value b_k (b_k') is set between the lowest (highest) value of the x_k belonging to the training vectors of the C_i class covered by this node and the x_k value of the nearest vector from another class. Those features that cover the whole data range are deleted, and for the remaining features further selection is done by checking the number of errors on vectors belonging to classes other than the class assigned to a given node. This procedure is repeated for all network nodes.

Optimization of rules

LOGICAL rules obtained from analysis of neural networks 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. 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.

Optimization of the intervals for linguistic variables at the network level is followed by maximization of the 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 $\mathbf{R}(C_i,C_j)$), $\min_M \left[\sum_{i\neq j} \mathbf{R}(C_i,C_j) P(C_i,C_j|M)\right]$. Using weighted combination:

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

allows to minimize over parameters M without constraints and explore the accuracy/rejection tradeoff. 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). Minimization of the cost function for each rule R separately over parameters used only in this rule is cheaper and frequently works as well as minimization of parameters for all rules simultaneously. As a result several sets of rules $R^{(k)}$ with increasing reliability and increasing rejection rate are obtained, i.e. covering smaller areas of the input space in which vectors from a single class dominate. In practical applications this is quite useful, since some classifications may be done with high degree of certainty, while the classification probability of those cases that fall in the region covered by less reliable set of rules $R^{(k+1)}$ and are rejected by the $R^{(k)}$ set may be estimated by looking at all training vectors that belong to the same category.

Although we have obtained very good results using this method [2] it has some drawbacks. First, global minimization techniques (simulated annealing and shuffled multisimplex method) to minimize the number of classification errors since for the crisp logical rules all probabilities are 0 or 1. Second, logical rules may be brittle since the intervals are not placed in an optimal position from generalization point of view. 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. To overcome the brittleness problem we have proposed [2] to add an input noise to the training data, optimize intervals using crossvalidation or search for the middle of the range of the interval values for which the number of classification errors does not change. Using the soft trapezoid form of rules in FSM or MLP2LN network leads to a specific form of fuzzy rules, but optimization of all parameters is difficult. The method presented below is simpler and may be used for large sets of logical rules produced by any method.

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 $x \in (a,b)$:

$$P(x \in (a,b)) = \frac{1}{2}\left[\mathrm{erf}\left(\frac{b-x}{s_x\sqrt{2}}\right) - \mathrm{erf}\left(\frac{a-x}{s_x\sqrt{2}}\right)\right] \quad (5)$$

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 $i = 1,\ldots,N$. 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 problem probabilities are calculated as:

$$P(x \in C) = \sum_{R \in 2^{R_C}} (-1)^{|R|+1} P(x \in \bigcap R) \quad (6)$$

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

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. An alternative possibility that we have considered[1], but not implemented yet, is to use the renormalized network outputs to compute probabilities:

$$p(C_k|X) = \frac{o_k(X)}{\sum_i o_i(X)} \quad (7)$$

with output neurons for class k summing the contributions of rule nodes,

$$o_k(X) = \sigma\left(\sum_i R_{i,k}(X)\right) \quad (8)$$

Each of these rule nodes computes normalized products of L-unit outputs connected to it. Although results from this approach are not equivalent to Monte Carlo simulations $p(C_k|X)$ values behave like probabilities and may be useful.

This approach to soft optimization may be used with any set of crisp logical rules to overcome the brittleness problem and to obtain robust wide margin rule-based classifiers. One should note that for large input uncertainties minimizing the mean square error (MSE) leads to different results than minimizing classification error. "Wide margins" are desirable if there is a gap between distribution of vectors belonging to different classes. If a single parameter s scaling all s_x is used it may be hard to avoid an increase of the number of classification errors despite the fact that the overall probability of correct classification will increase. To avoid this problem a few iterative steps are used: after minimization s is decreased and minimization repeated until s becomes sufficiently small and probabilities almost binary. In the limit minimization of MSE becomes equivalent to minimization of the classification error but the brittleness problem is solved because the intervals that are optimally placed for larger input uncertainties do not change in subsequent minimizations.

Applications

LARGE number of datasets were analyzed using an early version of this methodology, comparing results obtained from logical rules with results of other methods. Detailed comparison may be found on our Web page:
http://www.phys.uni.torun.pl/kmk/projects/rules.html.
Summary of our results is given in the Table 1. The classification accuracy in most cases is either better or comparable (within statistical error bars) to the best classifiers, and the number of logical rules lower than obtained by alternative systems. Each "or" condition and "else" condition is counted as a separate rule, therefore in two-class problem at least two rules are needed. Accuracy refers to the accuracy on the test set (if provided) or to the overall accuracy of rules on the whole dataset (if there is no test set). All data are from the UCI repository [11], except for the appendicitis, obtained from the authors of [12] paper. Hepatitis dataset contains many missing values and if averages are used meaningless rules are obtained; here only attributes with few missing values were used (no more than 5). NASA shuttle (described below) and the hypothyroid data contain large test sets, so comparison is independent of the details of crossvalidation. For the three monk problems – artificial data, not mentioned in the Table – perfect accuracy is obtained.

[1] We are grateful to Norbert Jankowski for this idea.

TABLE I
SUMMARY OF RULE EXTRACTION RESULTS. SECOND COLUMN GIVES THE NUMBER OF RULES/ATTRIBUTES USED.

Dataset	Rules/attrib.	Accuracy %
Appendicitis	3/2	91.5
Breast cancer	3/2	95.6
(Wisconsin)	6/5	99.0
Cancer (Ljubliana)	2/2	77.2
	4/4	78.0
Diabetes	2/2	75.0
Hepatitis	3/5	88.4
Heart (Cleveland)	4/3	85.5
Hypothyroid	3/5	99.4
Iris	3/1	95.3
	3/2	98.0
Mushrooms	2/1	98.5
	3/2	99.4
	4/4	99.9
	5/6	100.0

A. NASA Shuttle

The Shuttle dataset from NASA contains 9 continuos numerical attributes related to the positions of radiators in the Space Shuttle. There are 43500 training vectors and 14500 test vectors, divided into 7 classes in a very uneven way: about 80% from class 1 and only 6 examples from class 6 in the training set. This data has been used in the Stalog project [15], therefore accuracy of our rules may be compared with many other classification systems (Table 2).

TABLE II
SUMMARY OF RESULTS FOR THE NASA SHUTTLE DATASET.

Method	Train %	Test %
Linear discrimination	95.02	95.17
Logistic discrimination	96.07	96.17
MLP+BP	95.50	96.57
RBF	98.40	98.60
k-NN	–	99.56
C4.5 dec. tree	99.96	99.90
k-NN + feature sel.	–	99.95
NewID dec. tree	100.00	99.99
15 logical rules	99.98	99.97

We have used the FSM network with rectangular membership functions. Initialization of the network gives 7 nodes achieving already 88% accuracy. Increasing accuracy (using constructive learning algorithm) on the training set to 94%, 96% and 98% leads to a total of 15, 18 and 25 nodes and accuracies on the test set of 95.5%, 97.8% and 98.5%. Backpropagation network reached an accuracy of 95.5% on the training set. *k*-NN is very slow in this case, requiring all 43500 training vectors as reference for computing distances, reaching on the test set 99.56% but with feature selection improving to 99.95%. Using network optimization of the FSM rules described in the previous chapter 15 logical rules were generated. For example, for the third class rules are:

$$F2 \in [-188.43, -27.50] \wedge F9 \in [\,1, 74]$$
$$F2 \in [-129.49, -21.11] \wedge F9 \in [17, 76] \qquad (9)$$

The set of 15 rules makes only 3 errors on the training set (99.99% accuracy), leaving 8 vectors unclassified, and no errors on the test set but leaving 9 vectors unclassified (0.06%). After fuzzification of inputs only 3 errors and 5 unclassified vectors are obtained for the training and 3 vectors are unclassified and 1 error is made (although with probability of correct class close to 50%) for the test set. These results are much better than those obtained from the MLP or RBF networks (as reported in the Stalog project [15]) and comparable with the results of the best decision trees which work very well for this problem. It is interesting to note that in the Stalog project the NewID tree (descendant of the ID3 tree), which gave the best results here, has not been among the first 3 best methods for any other of the 22 datasets analyzed. Results of the C4.5 decision tree are already significantly worse. Our rule extraction approach has consistently been giving top results (cf. Table 1). Logical rules provide highly accurate and quite simple description of Shuttle dataset.

B. Psychometric data

Our rule extraction methods were applied to 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.

Psychologists require a rule based system because a detailed interpretation of the test, including description of personality type, should be assigned to each diagnosis. Our datasets contained up to 34 classes (normal, neurotic, drug addicts, schizophrenic, psychopaths, organic problems, malingerers etc.), determined by experts skilled in psychometric tests (it is hard to evaluate the accuracy of their evaluation). Our initial logical rules achieved about 93% accuracy on the whole set, increasing to over 95%-97% (depending on the dataset used) after some optimization and fuzzification. Assumption of the finite accuracy (about 1-5 units) of the procedure used for calculation of the values of each scale proved to be very

helpful, allowing for classification of vectors which were rejected by the crisp rules. On average 2.5 logical rules per class were derived (a total of 50-60 rules), involving between 2 and 9 attributes (out of 14). For most classes there were only a few errors and it is quite probable that they are due to the incorrect interpretation of psychograms by psychologists. Symptoms of organic problems and of schizophrenia are easily confused with symptoms belonging to other classes and most errors were indeed in these two classes. A typical rule has conditions referring to the values of scales, for example:

IF $Mk \in [42-46] \wedge It \in [48-51]$ THEN drug addiction

and therefore may easily be interpreted in a verbal way. Each rule has detailed interpretation associated with it by psychologists. Fuzzification 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.

Conclusions

OPTIMIZATION of crisp logical rules derived from neural methods leads to sets of simple and highly accurate rules. Neural approaches to extraction of logical rules from data frequently ignore the issues related to optimization and application of rules [3]. We have used optimization of linguistic variables and logical rules both at the level of network optimization and maximization of the predictive power of sets of rules. For Gaussian distribution of measurement errors estimation of classification probability using Monte Carlo procedure may be replaced by analytical calculation, leading to "soft trapezoid" fuzzification of the crisp membership functions and providing a small number of additional adaptive parameters. Sets of crisp logical rules may then be used to calculate probabilities (instead of giving a binary 0, 1 answers), making also classification of the rejected vectors possible.

Simplest logical description for several benchmark problems was obtained (Table 1). Recent improvements of our rule extraction and optimization methodology include better linguistic unit construction, optimization of predictive power of rules, direct optimization of FSM nodes, soft evaluation of probabilities and optimization of Gaussian uncertainties. These improvements allowed us to find simple rules for datasets that could not be analyzed earlier, such as diabetes [14], significantly higher accuracy for such complex databases as NASA Shuttle, or applications to difficult, real life psychometric data analysis. Since simplest data description is frequently the most useful one should always try to extract crisp logical rules. One reason for good performance of the rule-based systems is due to the excellent control of complexity of these classifiers. Overfitting of the data manifests with a large number of logical rules that cover only a few cases. Looking at the number of cases each rule handles an optimal number of parameters is easily determined. 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, such as fuzzy or neural classifiers. Finding a global optimum of the error function for these more sophisticated classification systems is usually more difficult than for the set of rules, and if soft transfer or membership functions are used this optimum may be different from optimal classification error.

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

[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 Communic. 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 (1998) 1-9

[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] W. Duch, R. Adamczak and K. Grąbczewski, *Neural optimization of linguistic variables and membership functions.* ICONIP'99, Perth, Australia, Nov. 1999 (in print)

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