Hybrid neural-global minimization method of logical rule extraction

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

May 18, 2000

Abstract

Methodology of extraction of optimal sets of logical rules using neural networks and global minimization procedures has been developed. Initial rules are extracted using density estimation neural networks with rectangular functions or multi-layered perceptron (MLP) networks trained with constrained backpropagation algorithm, transforming MLPs into simpler networks performing logical functions. A constructive algorithm called C-MLP2LN is proposed, in which rules of increasing specificity are generated consecutively by adding more nodes to the network. Neural rule extraction is followed by optimization of rules using global minimization techniques. Estimation of confidence of various sets of rules is discussed. The hybrid approach to rule extraction has been applied to a number of benchmark and real life problems with very good results.

Keywords: computational intelligence, neural networks, extraction of logical rules, data mining

1 Logical rules - introduction.

Why should one use logical rules if other methods of classification – machine learning, pattern recognition or neural networks – may be easier to use and give better results? Adaptive systems M_W, including the most popular neural network models, are useful classifiers that adjust internal parameters W performing vector mappings from the input to the output space. Although they may achieve high accuracy of classification the knowledge acquired by such systems is represented in a set of numerical parameters and architectures of networks in an incomprehensible way. Logical rules should be preferred over other methods of classification provided that the set of rules is not too complex and classification accuracy is sufficiently high. Surprisingly, in many applications simple rules proved to be more accurate and were able to generalize better than various machine and neural learning algorithms. Many statistical, pattern recognition and machine learning [1] methods of finding logical rules have been designed in the past. Neural networks are also used to extract logical rules and select classification features. Unfortunately systematic comparison of neural and machine learning methods is still missing. Many neural rule extraction methods have recently been reviewed and compared experimentally [2], therefore we will not discuss them here. Neural methods focus on analysis of parameters (weights and biases) of trained networks, trying to achieve high fidelity of performance, i.e. similar results of classification by extracted logical rules and by the original networks. Non-standard form of rules, such as M-of-N rules (M out of N antecedents should be true), fuzzy rules, or decision trees [1] are sometimes useful but in this paper we will consider only standard IF ... THEN propositional rules.

In classification problems propositional rules may take several forms. Very general form of such rule is: IF $X \in K^{(i)}$ THEN Class$(X) = C_i$, i.e. if X belongs to the cluster $K^{(i)}$ then its class is C_i =Class$(K^{(i)})$, the same as for all vectors in this cluster. If clusters overlap non-zero probability of classification $p(C_i|X;M)$ for several classes is obtained. This approach does not restrict the shapes of clusters used in logical rules, but unless the clusters are visualized in some way (a difficult task in highly dimensional feature spaces) it does not give more understanding of the data than any black box classifier. A popular simplification of the most general form of logical rules is to describe clusters using separable "membership" functions. This leads to fuzzy rules, for example in the form:

$$p(C_k|X;M) = \frac{\mu^{(k)}(X)}{\sum_i \mu^{(i)}(X)} \tag{1}$$

where

$$\mu^{(k)}(X) = \prod_i \mu_i^{(k)}(X_i) \tag{2}$$

and $\mu^{(k)}(X)$ is the value of the membership function defined for cluster k. Such context-dependent or cluster-dependent membership functions are used in the Feature Space Mapping (FSM) neurofuzzy system [3]. The flexibility of this approach depends on the choice of membership functions. Fuzzy logic classifiers use most frequently a few triangular membership functions per one input feature [4]. These functions do not depend on the region of the input space, providing oval decision borders, similar to Gaussian functions (cf. Fig.1). Thus fuzzy rules give decision borders that are not much more flexible than those of crisp rules. More important than softer decision borders is the ability to deal with oblique distribution of data by rotating some decision borders. This requires new linguistic variables formed by taking linear combination or making non-linear transformations of input features, but the meaning of such rules is sometimes difficult to comprehend (cf. proverbial "mixing apples with oranges"). Logical rules require symbolic inputs (linguistic variables), therefore the input data has to be quantized first, i.e. the features defining the problem should be identified and their values (sets of symbolic or integer values, or continuous intervals) labeled. For example a variable "size" will have the value "small" if the continuous variable x_k measuring size falls in some specified range, $x_k \in [a,b]$. Using one input variable several binary (logical) variables may be created, for example $s_1 = \delta(size, small)$ equal to 1 (true) only if variable "size" has the value "small".

The rough set theory [5] is used to derive crisp logic propositional rules. In this theory for two-class problems the lower approximation of the data is defined as a set of vectors or a region of the feature space containing input vectors that belong to a single class with probability $= 1$, while the upper approximation covers all instances which have a chance to belong to this class (i.e. probability is > 0). In practice the shape of the boundary between the upper and the lower approximations depends on the indiscernibility (or similarity) relation used. Linear approximation to the boundary region leads to trapezoidal membership functions. The simplest crisp form of logical rules is obtained if trapezoidal membership functions are changed into rectangular functions. Rectangles allow to define logical linguistic variables for each feature by intervals or sets of nominal values.

A fruitful way of looking at logical rules is to treat them as an approximation to the posterior probability of classification $p(C_i|X;M)$, where the model M is composed of the set of rules. Crisp, fuzzy and rough set decision borders are a special case of the FSM neurofuzzy approach [3] based on separable functions used to estimate the classification probability. Although the decision borders of crisp logical rule classification are much simpler than those achievable by neural networks, results

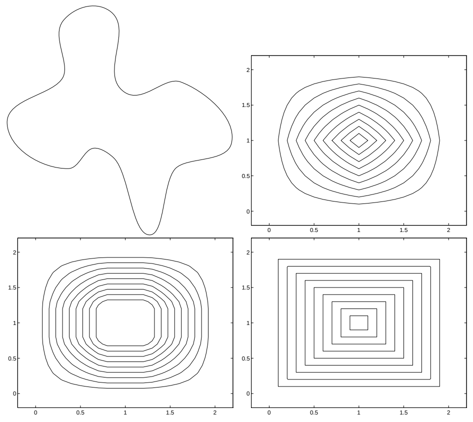
Figure 1: Shapes of decision borders for general clusters, fuzzy rules (using product of membership function), rough rules (trapezoidal approximation) and logical rules.

are sometimes significantly better. Three possible explanations of this empirical observation are: 1) the inability of soft sigmoidal functions to represent sharp, rectangular edges that may be necessary to separate two classes, 2) the problem of finding globally optimal solution of the non-linear optimization problem for neural classifiers – since we use a global optimization method to improve our rules there is a chance of finding a better solution than gradient-based neural classifiers are able to find; 3) the problem of finding an optimal balance between the flexibility of adaptive models and the danger of overfitting the data. Although Bayesian regularization [6] may help in case of some neural and statistical classification models, logical rules give much better control over the complexity of the data representation and elimination of outliers.

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. This became evident in the case of the "Hepar dataset", collected by dr. H. Wasyluk and co-workers from the Postgraduate Medical Center in Warsaw. The data contained 570 cases described by 119 values of medical tests and other features and were collected over a period of more than 5 years. These cases were divided into 16 classes, corresponding to different types of liver disease (final diagnosis was confirmed by analysis of liver samples under microscope, a procedure that we try to avoid by providing reliable diagnosis system). We have extracted crisp logical rules from this dataset using our C-MLP2LN approach described below, and found that 6 very simple rules gave 98.5% accuracy.

Unfortunately these rules also revealed that the missing attributes in the data were replaced by the averages for a given class, for example, cirrhosis was fully characterized by the rule "feature$_5$=4.39". Although one may report very good results using crossvalidation tests using this data, such data is of course useless since the averages for a given class are known only after the diagnosis.

In this paper a new methodology for logical rule extraction is presented, based on constructive MLPs for initial feature selection and rule extraction, and followed by rule optimization using global minimization methods. This approach is presented in detail in the next section and illustrative applications to a few benchmark problems are given in the third section. The fourth section contains applications to a real life medical data. The paper is finished with a short discussion.

2 The hybrid rule extraction methodology

Selection of initial linguistic variables (symbolic inputs) is the first step of the rule extraction process. Linguistic variables are optimized together with rules in an iterative process: neural networks with linguistic inputs are constructed, analysed, logical rules are extracted, intervals defining linguistic variables are optimized using rules, and the whole process repeated until convergence is achieved (usually two or three steps are sufficient, depending on the initial choice).

Linguistic (logical) input variables s_k, for integer or symbolic variables x_i, taking values from a finite set of elements $\mathcal{X}_i = \{X_i^{(j)}\}$, are defined as true if x_i belongs to a specific subset $\mathcal{X}_{ik} \subseteq \mathcal{X}_i$. Defining $\mathcal{X}$ as a subset of integers such linguistic variables as "prime-number" or "odd-number" may be defined. For continuous input features x_i linguistic variable s_k is obtained by specifying an open or closed interval $x_i \in [X_k, X'_k]$ or some other predicate $P_k(x_i)$. Each continuous input feature x_i is thus replaced by two or more linguistic values for which predicates P_k are true or false. For neural networks a convenient representation of these linguistic values is obtained using vectors of predicates, for example $V_{s_1} = (+1, -1, -1...)$ for linguistic variable s_1 or $V_{s_2} = (-1, +1, -1...)$ for s_2 etc.

Initial values of the intervals for continuous linguistic variables may be determined by the analysis of histograms, dendrograms, decision trees or clusterization methods. Neural-based methods are also suitable to select relevant features and provide initial intervals defining linguistic variables. FSM, our density estimation neurofuzzy network, is initialized using simple clusterization methods [7], for example dendrogram analysis of the input data vectors (for very large datasets rounding of continuous values to lower precision is applied first, and then duplicate data vectors are sorted out to reduce the number of input vectors). Rectangular functions may be used in FSM for a direct extraction of logical rules.

MLP (multilayered perceptron) neural models may find linguistic variables by training a special neural layer [8] of L-units (linguistic units). Each L-unit is a combination of two sigmoidal functions realizing the function $L(x_i; b_i, b'_i, \beta) = \sigma(\beta(x_i - b_i)) - \sigma(\beta(x_i - b'_i))$, parameterized by two biases b, b' determining the interval in which this function has non-zero value. The slope β of sigmoidal functions $\sigma(\beta x)$ is slowly increased during the learning process, transforming the fuzzy membership function ("soft trapezoid") into a window-type rectangular function [3, 8] (for simplicity of notation the dependence of L-functions on the slope is not shown below). Similar smooth transformation is used in the FSM network using biradial transfer functions, which are combinations of products of $L(x_i; b_i, b'_i)$ functions [9] with some additional parameters. Outputs of L-units $L(x_i; b_i, b'_i)$ are usually combined and filtered through another sigmoid $\sigma(\sum_{ij} L(x_i; b_{ij}, b'_{ij}))$ or the product $\prod_{ij} L(x_i; b_{ij}, b'_{ij}))$ of these functions is used.

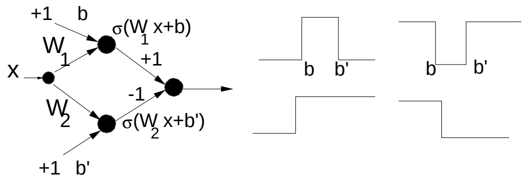

Figure 2: L-units, or pairs of neurons with constrained weights, used for determination of linguistic variables.

After initial definition of linguistic variables **neural network is constructed and trained**. To facilitate extraction of logical rules from an MLP network one should transform it smoothly into something resembling a network performing logical operations (Logical Network, LN). This transformation, called MLP2LN [10], may be realized in several ways. Skeletonization of a large MLP network is the method of choice if our goal is to find logical rules for an already trained network. Otherwise it is simpler to start from a single neuron and construct the logical network using the training data. Since interpretation of the activation of the MLP network nodes is not easy [11] a smooth transition from MLP to a logical-type of network performing similar functions is advocated. This is achieved by: a) gradually increasing the slope β of sigmoidal functions to obtain crisp decision regions; b) simplifying the network structure by inducing the weight decay through a penalty term; c) enforcing the integer weight values 0 and ±1, interpreted as 0 = irrelevant input, +1 = positive and -1 = negative evidence. These objectives are achieved by adding two additional terms to the standard mean square error function $E_0(W)$:

$$E(W) = E_0(W) + \frac{\lambda_1}{2}\sum_{i,j} W_{ij}^2 + \frac{\lambda_2}{2}\sum_{i,j} W_{ij}^2 (W_{ij}-1)^2 (W_{ij}+1)^2 \quad (3)$$

The first term, scaled by the λ_1 hyperparameter, encourages weight decay, leading to skeletonization of the network and elimination of irrelevant features. The second term, scaled by λ_2, forces the remaining weights to approach ± 1, facilitating easy logical interpretation of the network function. In the backpropagation training algorithm these new terms lead to the additional change of weights: $\lambda_1 W_{ij} + \lambda_2 W_{ij}(W_{ij}^2 - 1)(3W_{ij}^2 - 1)$. The last, 6-th order regularization term in the cost function, may be replaced by one of the lower order terms:

$$\begin{aligned} & |W_{ij}||W_{ij}^2 - 1| \text{ cubic} \quad (4)\\ & |W_{ij}| + |W_{ij}^2 - 1| \text{ quadratic} \\ & \sum_{k=-1}^{+1} |W_{ij} + k| - |W_{ij} - \frac{1}{2}| - |W_{ij} + \frac{1}{2}| - 1 \end{aligned}$$

Introduction of integer weights may also be justified from the Bayesian perspective [6]. The cost function specifies our prior knowledge about the probability distribution $P(W|M)$ of the weights in our model M. For classification task when crisp logical decisions are required the prior probability of the weight values should include not only small weights but also large positive and negative weights distributed around ± 1, for example:

$$P(W|M) = Z(\alpha)^{-1} e^{-\alpha E(W|M)} \propto \prod_{ij} e^{-\alpha_1 W_{ij}^2} \prod_{ij} e^{-\alpha_2 |W_{ij}^2 - 1|} \quad (5)$$

where the parameters α_i play similar role for probabilities as the parameters λ_i for the cost function. Using alternative cost functions amounts to different priors for regularization, for example Laplace instead of Gaussian prior. Initial knowledge about the problem may also be inserted directly into the network structure, defining initial conditions modified further in view of the incoming data. Since the final network structure becomes quite simple insertion of partially correct rules to be refined by the learning process is quite straightforward.

Although constraints Eq. (3) do not change the MLP exactly into a logical network they are sufficient to facilitate logical interpretation of the final network function. MLPs are trained using relatively large λ_1 value at the beginning, followed by a large λ_2 value near the end of the training process. These parameters determine the simplicity/accuracy tradeoff of the generated network and extracted rules. If a very simple network (and later logical rules) is desired, giving only rough description of the data, λ_1 should be as large as possible: although one may estimate the relative size of the regularization term versus the mean square error (MSE) a few experiments are sufficient to find the largest value for which the MSE is still acceptable and does not decrease quickly when λ_1 is decreased. Smaller values of λ_1 should be used to obtain more accurate networks (rules). The value of λ_2 should reach the value of λ_1 near the end of the training.

Logical rule extraction: the slopes of sigmoidal functions are gradually increased to obtain sharp decision boundaries and the complex decision regions are transformed into simpler, hypercuboidal decision regions. Rules $\mathcal{R}_k$ implemented by trained networks are obtained in the form of logical conditions by considering contributions of inputs for each linguistic variable s, represented by a vector V_s. Contribution of variable s to the activation is equal to the dot product $V_s \cdot W_s$ of the subset W_s of the weight vector corresponding to the V_s inputs. A combination of linguistic variables activating the hidden neuron above the threshold is a logical rule of the form: $\mathcal{R} = (s_1 \wedge \neg s_2 \wedge ... \wedge s_k)$.

In the **constructive version** of the MLP2LN approach (called C-MLP2LN) usually one hidden neuron per output class is created at a time and the training proceeds until the modified cost function reaches minimum. The weights and the threshold obtained are then analyzed and the first group of logical rules is found, covering the most common input-output relations. The input data that are correctly handled by the first group of neurons will not contribute to the error function, therefore the weights of these neurons are kept frozen during further training. This is equivalent to training one neuron (per class) at a time on the remaining data, although sometimes training two or more neurons may lead to faster convergence. This procedure is repeated until all the data are correctly classified, weights analyzed and a set of rules $\mathcal{R}_1 \vee \mathcal{R}_2 ... \vee \mathcal{R}_n$ is found for each output class or until the number of rules starts to grow rapidly. The output neuron for a given class is connected to the hidden neurons created for that class – in simple cases only one neuron may be sufficient to learn all instances, becoming an output neuron rather than a hidden neuron. Output neurons perform simple summation of the hidden nodes signals. Since each time only one neuron per class is trained the C-MLP2LN training is fast. The network repeatedly grows when new neurons are added, and then shrinks when connections are deleted. Since the first neuron for a given class is trained on all data for that class the rules it learns are most general, covering largest number of instances. Therefore rules obtained by this algorithm are ordered, starting with rules that cover many cases and ending with rules that cover only a few cases.

Simplification of rules: the final solution may be presented as a set of logical rules or as a network of nodes performing logical functions. However, 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. Therefore after extraction rules are carefully analyzed and whenever possible simplified (we use a Prolog program for this step).

Optimization of rules: optimal intervals and other adaptive parameters may be found by maximization of a predictive power of a classifier. Let $\mathbf{P}(C_i, C_j|M)$ 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 some parameters M. Then for n samples $\mathbf{p}(C_i, C_j|M) = \mathbf{P}(C_i, C_j|M)/n$ is the probability of (mis)classification. The best parameters M are selected by maximizing the number (or probability) of correct predictions (called also the "predictive power" of rules):

$$\max_M \left[\text{Tr}\, \mathbf{P}(C_i, C_j|M)\right] \tag{6}$$

over all parameters M, or minimizing the number of wrong predictions (possibly with some risk matrix $\mathbf{R}(C_i, C_j)$):

$$\min_M \left[\sum_{i \neq j} \mathbf{R}(C_i, C_j)\mathbf{P}(C_i, C_j|M)\right] \tag{7}$$

Weighted combination of these two terms:

$$E(M) = \lambda \sum_{i \neq j} \mathbf{P}(C_i, C_j|M) - \text{Tr}\, \mathbf{P}(C_i, C_j|M) \tag{8}$$

is bounded by $-n$ and should be minimized over parameters M without constraints. For this minimization we have used simulated annealing or multisimplex global minimization methods. If λ 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). Since rules discriminate between instances of one class and all other classes one can define a cost function for each rule separately:

$$E_R(M) = \lambda(\mathbf{P}_{+-} + \mathbf{P}_{-+}) - (\mathbf{P}_{++} + \mathbf{P}_{--}) \tag{9}$$

and minimize it over parameters M used in the rule $\mathcal{R}$ only ($+$ means here one of the classes, and $-$ means all other classes). The combination $\mathbf{P}_{++}/(\mathbf{P}_{++} + \mathbf{P}_{+-}) \in (0, 1]$ is sometimes called the sensitivity of a rule [12], while $\mathbf{P}_{--}/(\mathbf{P}_{--} + \mathbf{P}_{-+})$ is called the specificity of a rule. Some rule induction methods optimize such combinations of $\mathbf{P}_{x,y}$ values.

Estimation of the accuracy of rules is very important, especially in medicine. Tests of classification accuracy should be performed using stratified 10-fold cross-validation, each time including rule optimization on the training set. Changing the value of λ_1 will produce a series of models with higher and higher confidence of correct classification at the expense of growing rejection rate. A set of rules may classify some cases at the 100% confidence level; if some instances are not covered by this set of rules another set of (usually simpler) rules at a lower confidence level is used (confidence level is estimated as the accuracy of rules achieved on the training set). In this way a reliable estimation of confidence is possible. The usual procedure is to give only one set of rules, assigning to each rule a confidence factor, for example $c_i = P(C_i, C_i|M)/\sum_j P(C_i, C_j|M)$. This is rather misleading. A rule $\mathcal{R}^{(1)}$ that does not make any errors covers typical instances and its reliability is close to 100%. If a less accurate rule $\mathcal{R}^{(2)}$ is given, for example classifying correctly 90% of instances, the reliability of classification for instances covered by the first rule is still close to 100% and the reliability of classification in the border region ($\mathcal{R}^{(2)} \backslash \mathcal{R}^{(1)}$, or cases covered by $\mathcal{R}^{(2)}$ but not by $\mathcal{R}^{(1)}$) is much less than 90%. Including just these border cases gives much lower confidence factors and since the number of such cases is relatively small the estimate itself has low reliability. A possibility sometimes worth considering is to use a similarity-based classifier (such as the k-NN method or RBF network) to improve accuracy in the border region.

Logical rules, similarly as any other classification systems, may become brittle if the decision borders are placed too close to the data vectors instead of being placed between the clusters. **The brittleness problem** is solved either at the optimization stage by selecting the middle values of the intervals for which best performance is obtained or, in a more general way, by adding noise to the data [13]. So far we have used the first method: starting from the values of the optimized parameters the largest cuboid in the parameter space in which the number of errors is constant is determined. The center of this cuboid is taken as the final estimation of the adaptive parameters.

3 Illustrative benchmark applications.

Datasets for benchmark applications were taken from the UCI machine learning repository [14]. Application of the constructive C-MLP2LN approach to the classical Iris dataset was already presented in detail [15], therefore only new aspects related to the hybrid method are discussed here. The Iris data has 150 vectors evenly distributed in three classes: iris-setosa, iris-versicolor and iris-virginica. Each vector has 4 features: sepal length x_1 and width x_2, and petal length x_3 and width x_4 (all in cm). Analysis of smoothed histograms (assuming Gaussian width for each value) of the individual features for each class provides initial linguistic variables. Assuming at most 3 linguistic variables per input feature a network with 12 binary inputs equal to ± 1 (features present or absent) is constructed. For example, the medium value of a single feature is coded by $(-1, +1, -1)$ vector. For the Iris dataset a single neuron per one class was sufficient to train the network, therefore the final network structure (Fig. 3) has 12 input nodes and 3 output nodes (hidden nodes are only needed when more than one neuron is necessary to cover all the rules for a given class).

The constraint hyperparameters were increased from $\lambda_1 = 0.001$ at the beginning of the training to about $\lambda_1 = 0.1$ near the end, with λ_2 increasing from 0 to 0.1 to enforce integer weights near the end of the training. On average the network needed about 1000 epochs for convergence. The final weights are taken to be exactly ± 1 or 0 while the final value of the slopes of sigmoids reaches 300. Using L-units with initial linguistic variables provided by the histograms very simple networks were created, with non-zero weights for only one attribute, petal length x_3. Two rules and else condition, giving overall 95.3% accuracy (7 errors) are obtained:

$\mathcal{R}_1^{(1)}$: iris-setosa if $x_3 < 2.5$ (100%),
$\mathcal{R}_2^{(1)}$: iris-virginica if $x_3 > 4.8$ (92%),
$\mathcal{R}_3^{(1)}$: else iris-versicolor (94%)

The first rule is accurate in 100% of cases since the setosa class is easily separated from the two other classes. Lowering the final hyperparameters leads to the following weights and thresholds (only the signs of the weights are written):

Setosa	(0,0,0	0,0,0	+,0,0	+,0,0)	$\theta = 1$
Versicolor	(0,0,0	0,0,0	0,+,-	0,+,-)	$\theta = 3$
Virginica	(0,0,0	0,0,0	-,-,+	-,-,+)	$\theta = 2$

Interpretation of these weights and the resulting network function (Fig. 3) is quite simple. Only two features, x_3 and x_4, are relevant and a single rule per class is found:

setosa if $(x_3 < 2.9 \ \vee \ x_4 < 0.9)$ (100%)
versicolor if $(x_3 \in [2.9, 4.95] \wedge x_4 \in [0.9, 1.65])$ (100%)
virginica if $(x_3 > 4.95) \ \vee \ (x_4 > 1.65)$ (94%)

This $\mathcal{R}^{(2)}$ level set of rules allows for correct classification of 147 vectors, achieving overall 98.0% accuracy. However, the first two rules have 100% reliability while all errors are due to the third rule, covering 53 cases. Decreasing constraint hyperparameters further allows to replace one of these rules by four rules, with a total of three attributes and 11 antecedents, necessary to classify correctly a single additional vector, a clear indication that overfitting occurs.

The third set of rules $\mathcal{R}^{(3)}$ has been found after optimization with increasing λ (Eq. 8), reliable in 100% but rejecting 11 vectors, 8 virginica and 3 versicolor:

setosa if $(x_3 < 2.9)$ (100%)
versicolor if $(x_3 \in [2.9, 4.9] \ \wedge x_4 < 1.7)$ (100%)
virginica if $(x_3 \geq 5.3 \ \vee x_4 \geq 1.9)$ (100%)

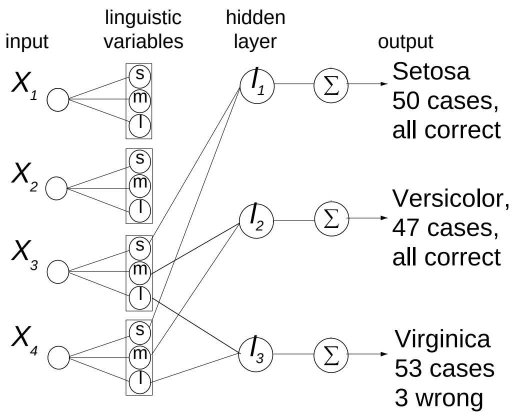

Figure 3: Final structure of the network for the Iris problem.

The reliability of classification for the 11 vectors in the border region ($\mathcal{R}^{(2)} \setminus \mathcal{R}^{(3)}$) is rather low: with $p = 8/11$ they should be assigned to the virginica class and with $p = 3/11$ to the versicolor class. It is possible to generate more specific rules, including more features, just for the border region, or to use in this region similarity-based classification system, such as k-NN, but for this small dataset we do not expect any real improvement since the true probability distributions of leave's sizes for the two classes of iris flowers clearly overlap.

In the **mushroom problem** [2, 14] the database consists of 8124 vectors, each with 22 discrete attributes with up to 10 different values. 51.8% of the cases represent edible, and the rest non-edible (mostly poisonous) mushrooms. A single neuron is capable of learning all the training samples (the problem is linearly separable), but the resulting network has many nonzero weights and is difficult to analyze from the logical point of view. Using the C-MLP2LN algorithm with the cost function Eq. (3) the following disjunctive rules for poisonous mushrooms have been discovered:

$\mathcal{R}_1$) odor=$\neg$(almond$\vee$anise$\vee$none)
$\mathcal{R}_2$) spore-print-color=green
$\mathcal{R}_3$) odor=none$\wedge$stalk-surface-below-ring=scaly$\wedge$
(stalk-color-above-ring=$\neg$brown)
$\mathcal{R}_4$) habitat=leaves$\wedge$cap-color=white

Rule $\mathcal{R}_1$ misses 120 poisonous cases (98.52% accuracy), adding rule $\mathcal{R}_2$ leaves 48 errors (99.41% accuracy), adding third rule leaves only 8 errors (99.90% accuracy), and all rules $\mathcal{R}_1$to $\mathcal{R}_4$ classify all poisonous cases correctly. The first two rules are realized by one neuron. For large value of the weight-decay parameter only one rule with odor attribute is obtained, while for smaller hyperparameter values a second attribute (spore-print-color) is left. Adding a second neuron and training it on the remaining cases generates two additional rules, R_3 handling 40 cases and R_4 handling only 8 cases. We have also derived the same rules using only 10% of all data for training. This is the simplest systematic logical description of the mushroom dataset that we know of (some of these rules have probably been also found by the RULEX and TREX algorithms [2]) and therefore should be used as a benchmark for other rule extraction methods.

We have also solved the three monk problems [16]. For the Monk 1 problem a total of 4 rules and one exception classifying the data without any errors was created (exceptions are additional rules handling patterns that are not recognized properly by the rules). In the Monk 2 problem perfect classification with 16 rules and 8 exceptions extracted from the resulting network has been achieved. The number of atomic formulae which compose them is 132. In the Monk 3 problem, although the training data for this problem is corrupted by 5% noise it is still possible to obtain 100% accuracy [2]. The whole logical system for this case contains 33 atomic formulae.

4 Applications to real-world medical data

To facilitate comparison with results obtained by several classification methods we have selected three well-known medical datasets obtained from the UCI repository [14].

4.1 Wisconsin breast cancer data.

The Wisconsin cancer dataset [17] contains 699 instances, with 458 benign (65.5%) and 241 (34.5%) malignant cases. Each instance is described by the case number, 9 attributes with integer value in the range 1-10 (for example, feature f_2 is “clump thickness” and f_8 is “bland chromatin”) and a binary class label. For 16 instances one attribute is missing. This data has been analyzed in a number of papers (Table 1).

The simplest rules obtained from optimization includes only two rules for malignant class:

$$f_2 \geq 7 \vee f_7 \geq 6 \qquad (95.6\%)$$

These rules cover 215 malignant cases and 10 benign cases, achieving overall accuracy (including ELSE condition) of 94.9%. Without optimization 5 disjunctive rules were initially obtained from the C-MLP2LN procedure for malignant cases, with benign cases covered by the ELSE condition:

$\mathcal{R}_1$: $f_2 < 6 \wedge f_4 < 4 \wedge f_7 < 2 \wedge f_8 < 5$ (100)%
$\mathcal{R}_2$: $f_2 < 6 \wedge f_5 < 4 \wedge f_7 < 2 \wedge f_8 < 5$ (100)%
$\mathcal{R}_3$: $f_2 < 6 \wedge f_4 < 4 \wedge f_5 < 4 \wedge f_7 < 2$ (100)%
$\mathcal{R}_4$: $f_2 \in [6,8] \wedge f_4 < 4 \wedge f_5 < 4 \wedge f_7 < 2 \wedge f_8 < 5$ (100)%
$\mathcal{R}_5$: $f_2 < 6 \wedge f_4 < 4 \wedge f_5 < 4 \wedge f_7 \in [2,7] \wedge f_8 < 5$ (92.3)%

The first 4 rules achieve 100% accuracy (i.e. they cover cases of malignant class only), the last rule covers only 39 cases, 36 malignant and 3 benign. The confusion matrix is: $P = \begin{pmatrix} 238 & 3 \\ 25 & 433 \end{pmatrix}$, i.e. there are 3 benign cases wrongly classified as malignant and 25 malignant cases wrongly classified as benign, giving overall accuracy of 96%. Optimization of this set of rules gives:

$\mathcal{R}_1$: $f_2 < 6 \wedge f_4 < 3 \wedge f_8 < 8$ (99.8)%
$\mathcal{R}_2$: $f_2 < 9 \wedge f_5 < 4 \wedge f_7 < 2 \wedge f_8 < 5$ (100)%
$\mathcal{R}_3$: $f_2 < 10 \wedge f_4 < 4 \wedge f_5 < 4 \wedge f_7 < 3$ (100)%
$\mathcal{R}_4$: $f_2 < 7 \wedge f_4 < 9 \wedge f_5 < 3 \wedge f_7 \in [4,9] \wedge f_8 < 4$ (100)%
$\mathcal{R}_5$: $f_2 \in [3,4] \wedge f_4 < 9 \wedge f_5 < 10 \wedge f_7 < 6 \wedge f_8 < 8$ (99.8)%

These rules classify only 1 benign vector as malignant ($\mathcal{R}_1$ and $\mathcal{R}_5$, the same vector), and the ELSE condition for the benign class makes 6 errors, giving 99.00% overall accuracy of this set of rules. Minimizing Eq. (8) we have also obtained 7 rules (3 for malignant and 4 for benign class) working with 100% reliability but rejecting 79 cases (11.3%). In all cases features f_3 and f_6 (both related to the cell size) were not important and f_2 with f_7 were the most important. Incidentally, the Ljubliana cancer data [14] gives about 77% accuracy in crossvalidation tests using just a single logical rule.

Method	Accuracy %
IncNet [18]	97.1
LVQ	96.6
MLP+backprop	96.7
CART (decision tree)	94.2
LFC, ASI, ASR decision trees	94.4-95.6
Fisher LDA	96.8
Linear Discriminant Analysis	96.0
Quadratic Discriminant Analysis	34.5
3-NN, Manhattan (our results)	97.0±0.12
Naive Bayes	96.4
Bayes (pairwise dependent)	96.6
FSM (our results)	96.5

Table 1: Results from the 10-fold crossvalidation for the Wisconsin breast cancer dataset, data from [19] or our own calculations, except IncNet.

4.2 The Cleveland heart disease data.

The Cleveland heart disease dataset [14] (collected at V.A. Medical Center, Long Beach and Cleveland Clinic Foundation by R. Detrano) contains 303 instances, with 164 healthy (54.1%) instances, the rest are heart disease instances of various severity. While the database has 76 raw attributes, only 13 of them are actually used in machine learning tests, including 6 continuous features and 4 nominal values. There are many missing values of the attributes. Results obtained with various methods for this data set are collected in Table 2.

After some simplifications the derived rules obtained by the C-MLP2LN approach are:

$\mathcal{R}_1$: (thal=0 $\vee$ thal=1) $\wedge$ ca=0.0 (88.5%)
$\mathcal{R}_2$: (thal=0 $\vee$ ca=0.0) $\wedge$ cp$\neq$ 2 (85.2%)

These rules give 85.5% correct answers on the whole set and compare favorable with the accuracy of other classifiers (Table 2).

Method	Accuracy %
LVQ	82.9
MLP+backprop	81.3
CART (decision tree)	80.8
LFC, ASI, ASR decision trees	74.4-78.4
Fisher LDA	84.2
Linear Discriminant Analysis	84.5
Quadratic Discriminant Analysis	75.4
k-NN	81.5
Naive Bayes	83.4
Bayes (pairwise dependent)	83.1
FSM - Feature Space Mapping	84.0

Table 2: Results from the 10-fold crossvalidation for the Cleveland heart disease dataset.

4.3 The hypothyroid data.

This is a somewhat larger dataset [14], with 3772 cases for training, 3428 cases for testing, 22 attributes (15 binary, 6 continuous), and 3 classes: primary hypothyroid, compensated hypothyroid and normal (no hypothyroid). the class distribution in the training set is 93, 191, 3488 vectors and in the test set 73, 177, 3178. For the first class two rules are sufficient (all values of continuous features are multiplied here by 1000):

$\mathcal{R}_1$: FTI $< 63\wedge$ TSH ≥ 29
$\mathcal{R}_2$: FTI $< 63\wedge$ TSH $\in [6.1, 29)\wedge$ T3< 20

For the second class one rule is created:

$\mathcal{R}_3$: FTI $\in [63, 180]\wedge$ TSH $\geq 6.1\wedge$on thyroxine=no $\wedge$ surgery=no

and the third class is covered by ELSE. With these rules we get 99.68% accuracy on the training set and 99.07% error on the test set. Optimization of the rules leads to slightly more accurate rules:

$\mathcal{R}_1$: TSH $\geq 30.48\wedge$ FTI < 64.27 (97.06%)
$\mathcal{R}_2$: TSH $\in [6.02, 29.53]\wedge$ FTI $< 64.27\wedge$ T3< 23.22 (100%)
$\mathcal{R}_3$: TSH $\geq 6.02\wedge$ FTI $\in [64.27, 186.71]\wedge$ TT4$< 150.5\wedge$ on thyroxine=no $\wedge$ surgery=no (98.96%)

The ELSE condition has 100% reliability (on the training set). These rules make only 4 errors on the training set (99.89%) and 22 errors on the test set (99.36%). They are similar to those found using heuristic version of PVM method by Weiss and Kapouleas [20]. The differences among PVM, CART and C-MLP2LN are for this dataset rather small (Table 3), but other methods, such as well-optimized MLP (including genetic optimization of network architecture) or cascade correlation classifiers, give results that are significantly worse. Poor results of k-NN are especially worth noting, showing that in this case, despite large amount of reference vectors, similarity-based methods are not competitive.

Method	% train	% test
k-NN [20]	–	95.3
Bayes [20]	97.0	96.1
3-NN, 3 features used	98.7	97.9
MLP+backprop [21]	99.60	98.45
Cascade correl. [21]	100.00	98.48
PVM [20]	99.79	99.33
CART [20]	99.79	99.36
C-MLP2LN	99.89	99.36

Table 3: Results for the hypothyroid dataset.

4.4 Psychometric data

The rule extraction and optimization approach described in this paper is used by us in several real-life projects. One of these projects concerns the psychometric data collected in the Academic Psychological Clinic of our University. A computerized version of the Minnesota Multiphasic Personality Inventory (MMPI) test was used, consisting of 550 questions with 3 possible answers each. MMPI evaluates psychological characteristics reflecting social and personal maladjustment, including psychological dysfunction. Hundreds of books and papers were written on the interpretation of this test (cf. review [22]). Many computerized versions of the MMPI test exist to assist in information acquisition, but evaluation of results is still done by an experienced clinical psychologist. Our goal is to provide automatic psychological diagnosis.

The raw MMPI data is used to compute 14 coefficients forming a psychogram. First four coefficients are just the control scales (measuring consistency of answers, allowing to find malingerers), with the rest forming clinical scales. They were developed to measure tendencies towards hypochondria, depression, hysteria, psychopathy, paranoia, schizophrenia etc. A large number of simplification schemes has been developed to make the interpretation of psychograms easier. They may range from rule-based systems derived from observations of characteristic shapes of psychograms, Fisher discrimination functions, or systems using a small number of coefficients, such as the 3 Goldberg coefficients. Unfortunately there is no comparison of these different schemes and their relative merits have not been tested statistically.

At present we have 1465 psychograms, each classified into one of 34 types (normal, neurotic, alcoholics, schizophrenic, psychopaths, organic problems, malingerers etc.) by an expert psychologist. This classification is rather difficult and may contain errors. Our initial logical rules achieve about 93% accuracy on the whole set. For most classes there are only a few errors and it is quite probable that they are due to the errors of the psychologists interpreting the psychogram data. The only exception is the class of organic problems, which leads to answers that are frequently confused with symptoms belonging to other classes. On average 2.5 logical rules per class were derived, involving between 3 and 7 features. A typical rule has the form:

If $f_7 \in [55, 68] \wedge f_{12} \in [81, 93] \wedge f_{14} \in [49, 56]$ Then Paranoia

After optimization these rules will be used in an expert system and evaluated by clinical psychologist in the near future.

5 Summary

A new methodology for extraction of logical rules from data has been presented. Neural networks - either density estimation (FSM) or constrained multilayered perceptrons (MLPs) - are used to obtain initial sets of rules. FSM is trained either directly with the rectangular functions or making a smooth transition from biradial functions [9] or trapezoidal functions to rectangular functions. MLPs are trained with constraints that change them into networks processing logical functions, either by simplification of typical MLPs or by incremental construction of networks performing logical functions. In this paper we have used mostly the C-MLP2LN constructive method since it requires less experimentation with various network structures.

The method of successive regularizations introduced by Ishikawa [23] has several features in common with our MLP2LN approach and is capable (although in our opinion it requires more effort to select the initial network architecture) of producing similar results [24]. Specific form of the cost function as well as the C-MLP2LN constructive algorithm in which neurons are added and then connections are deleted, seems to be rather different from other algorithms used for logical rule extraction so far [2]. Neural and machine learning methods should serve only for feature selection and initialization of sets of rules, with final optimization done using global minimization (or search) procedures. Optimization leads to rules that are more accurate and simpler, providing in addition sets of rules with different reliability.

Using C-MLP2LN hybrid methodology simplest logical description for several benchmark problems (Iris, mushroom) has been found and perfect solutions were obtained for the three monk problems. For many medical datasets (only 3 were shown here) very simple and highly accurate results were obtained. It is not quite clear why logical rules work so well, for example in the hypothyroid or the Wisconsin breast cancer case obtaining accuracy which is better than that of any other classifier. One 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. Logical rules given in this paper were actually used by us to initialize MLPs but high accuracy is preserved only with extremely steep slopes of sigmoidal functions. In any case the approach presented here is ready to be used in real world applications and we are at present applying it to complex medical and psychometric data.

Acknowledgments

Support by the Polish Committee for Scientific Research, grant 8 T11F 014 14, is gratefully acknowledged. We would like to thank J. Gomuła and T. Kucharski for providing the psychometric data.

References

[1] T. Mitchell, “Machine learning”. McGraw Hill 1997

[2] 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, 373–389, 1995

[3] W. Duch and G.H.F. Diercksen, “Feature Space Mapping as a universal adaptive system”, Computer Physics Communications, 87: 341–371, 1995; W. Duch, R. Adamczak and N. Jankowski,

"New developments in the Feature Space Mapping model", 3rd Conf. on Neural Networks and Their Applications, Kule, Poland, October 1997, pp. 65-70

[4] N. Kasabov, "Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering", The MIT Press (1996).

[5] Z. Pawlak, "Rough sets - theoretical aspects of reasoning about data", Kluver Academic Publishers 1991

[6] D.J. MacKay. "A practical Bayesian framework for backpropagation networks", Neural Computations 4, 448-472, 1992

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

[8] W. Duch, R. Adamczak and K. Grąbczewski, "Constrained backpropagation for feature selection and extraction of logical rules", in: *Proc. of "Colloquiua in AI"*, Łódź, Poland 1996, p. 163–170

[9] W. Duch and N. Jankowski, "New neural transfer functions". Applied Mathematics and Computer Science 7, 639-658 (1997)

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

[11] J.M. Żurada. "Introduction to Artificial Neural Systems", West Publishing Co., St Paul, 1992.

[12] S.M. Weiss, C.A. Kulikowski, "Computer systems that learn", Morgan Kauffman, San Mateo, CA 1990

[13] C.M. Bishop, "Training with noise is equivalent to Tikhonov regularization", Neural Computation 7, 108-116 (1998)

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

[15] W. Duch, R. Adamczak, K. Grąbczewski, "Extraction of logical rules from training data using backpropagation networks", The 1st Online Workshop on Soft Computing, 19-30.Aug.1996; http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/, pp. 25-30

[16] W. Duch, R. Adamczak, K. Grąbczewski, "Extraction of crisp logical rules using constrained backpropagation networks", Int. Conf. on Artificial Neural Networks (ICNN'97), Houston, 9-12.6.1997, pp. 2384-2389

[17] K. P. Bennett, O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets", Optimization Methods and Software 1, 1992, 23-34.

[18] N. Jankowski N and V. Kadirkamanathan, "Statistical control of RBF-like networks for classification", 7th Int. Conf. on Artificial Neural Networks (ICANN'97), Lausanne, Switzerland, 1997, pp. 385-390

[19] B. Ster and A. Dobnikar, "Neural networks in medical diagnosis: Comparison with other methods". In: A. Bulsari et al., eds, Proc. Int. Conf. EANN'96, pp. 427-430, 1996.

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

[21] W. Schiffman, M. Joost, R. Werner, "Comparison of optimized backpropagation algorithms", Proc. of ESANN'93, Brussels 1993, pp. 97-104

[22] J.N. Butcher, S.V. Rouse, "Personality: individual differences and clinical assessment". Annual Review of Psychology 47, 87 (1996)

[23] M. Ishikawa, "Rule extraction by succesive regularization", in: Proc. of the 1996 IEEE ICNN, Washington, June 1996, pp. 1139–1143

[24] W. Duch, R. Adamczak, K. Grąbczewski, M. Ishikawa, H. Ueda, "Extraction of crisp logical rules using constrained backpropagation networks - comparison of two new approaches", Proc. of the European Symposium on Artificial Neural Networks (ESANN'97), Bruge 16-18.4.1997, pp. 109-114