KYBERNETIKA — VOLUME 41 (2005), NUMBER 3, PAGES 297 - 314

EXTRACTION OF FUZZY LOGIC RULES FROM DATA BY MEANS OF ARTIFICIAL NEURAL NETWORKS[1]

MARTIN HOLEŇA

The extraction of logical rules from data has been, for nearly fifteen years, a key application of artificial neural networks in data mining. Although Boolean rules have been extracted in the majority of cases, also methods for the extraction of fuzzy logic rules have been studied increasingly often. In the paper, those methods are discussed within a five-dimensional classification scheme for neural-networks based rule extraction, and it is pointed out that all of them share the feature of being based on some specialized neural network, constructed directly for the rule extraction task. As an important representative, a method for the extraction of rules in a general fuzzy disjunctive normal form is described in detail and illustrated on real-world applications. Finally, the paper proposes an algorithm demonstrating a principal possibility to extract fuzzy logic rules from multilayer perceptrons with continuous activation functions, i. e., from the kind of neural networks most universally used in applications. However, complexity analysis of the individual steps of that algorithm reveals that it involves computations with doubly-exponential complexity, due to which it can not without simplifications serve as a practically applicable alternative to methods based on specialized neural networks.

Keywords: knowledge extraction from data, artificial neural networks, fuzzy logic, Łukasiewicz logic, disjunctive normal form

AMS Subject Classification: 03B52, 62-07, 62M45, 68T30

1. INTRODUCTION

Extraction of logical rules from data is the main stream of the nowadays quite popular data mining technology, an information technology attempting to extract, from unmanageable and always increasing amounts of available data, manageable amounts of human-understandable structured knowledge. Knowledge expressible as logical rules has been traditionally extracted not only with purely logical approaches, but also with many statistically-based approaches and approaches relying on artificial neural networks and on nonconnectionist machine learning (Figure 1). In addition, rules are extracted also with the emerging data mining approaches based

[1] Presented at the 7th FSTA international conference held in Liptovský Mikuláš, Slovakia, on January 26–30, 2004.

on rough sets and on genetic algorithms. Although the notion of a rule is understood differently in different approaches, it always means some kind of formulas or sentences of some formal logic, typically some kind of implications or equivalences. The most frequently encountered rule extraction methods have been the topic of a number of specialized monographs (e. g., [1, 13, 16, 21, 24, 36, 48, 49, 53, 55, 56]). The present paper, which is an extended written version of a talk at the Seventh International Conference on Fuzzy Systems Theory and Applications, deals with a less known kind of rule extraction methods – methods for the extraction of fuzzy logic rules from data by means of artificial neural networks.

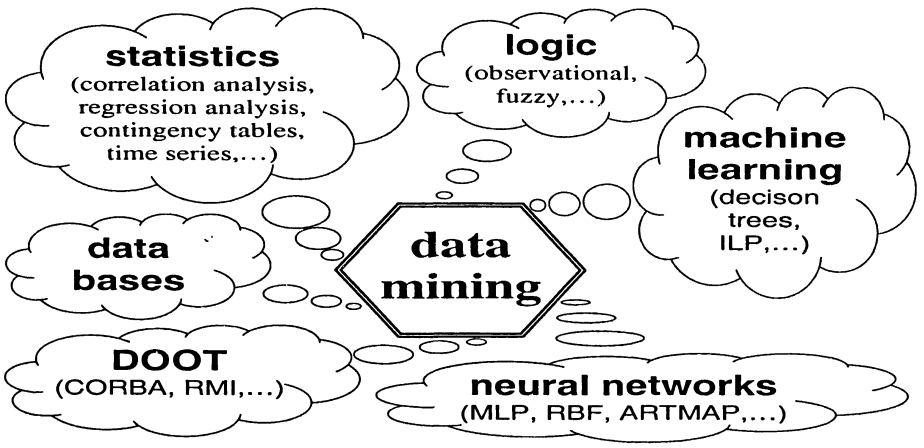

Fig. 1. Main data mining approaches and supporting technologies.

The following section presents a general characterization of neural-networks based rule extraction methods. Attention is paid mainly to their universality with respect to the underlying neural networks, and it is recalled that for the extraction of fuzzy rules, so far only specialized networks have been used, built up directly for the extraction of some particular kind of fuzzy logic rules. An example method relying on such a network is given in Section 3. Finally, Section 4 shows the principal possibility but unacceptably high computational complexity of extracting fuzzy rules from the kind of neural networks most frequently encountered in applications – multilayer perceptrons.

2. EXTRACTION OF LOGICAL RULES BY MEANS OF ARTIFICIAL NEURAL NETWORKS

The extraction of knowledge from data by means of artificial neural networks (ANNs) has received much attention especially in the nineties [3, 6, 31, 35, 38, 39, 51]. Actually, already the mapping computed by the network incorporates knowledge transferred to it during training from the training data, knowledge about the implications that certain values of the variables assigned to its inputs have for the values of the

variables assigned to its outputs. That knowledge is represented partially through the network architecture and mainly through distributed numerical parameters determining the computed mapping. Needless to say, such a knowledge representation is not easily human-comprehensible (in terms of [8], that representation provides a high *data fit* but a low *mental fit*). It is the difficult comprehensibility of such a representation that motivated research into the problem of extracting from it more easily comprehensible logical rules. Formally, that problem can be viewed as a *transformation of the ANN architecture and of parameters determining the computed mapping into a set of logical rules of a prescribed kind,* e. g., into a set of appropriate implications or equivalences [11, 14, 15, 20, 25, 30, 34, 50, 52, 54].

Up to now, already several dozens ANN-based rule extraction methods exist. In [6] and [51], a classification scheme has been proposed, classifying each such method according to following aspects:

(i) *expressive power* of the extracted rules, given by the meaning they are able to convey;

(ii) *translucency* of the view of the underlying neural network, i. e., the extent to which the extracted rules reflect the way how the mapping computed by the network is composed from somatic and synaptic mappings assigned to individual neurons and connections;

(iii) *universality* of the method with respect to how commonly used is the underlying neural network, which in turn determines its portability across networks encountered in various applications;

(iv) *quality of the set of extracted rules*, determined mainly by its *comprehensibility, consistency* and *completeness,* and by the *accuracy* and *fidelity* of the individual rules;

(v) *computational complexity* of the method.

The key aspect of that classification is the expressive power of the rules. Though the conveyable meaning of the rules depends also on the syntax of the language underlying the considered logic, which allows to differentiate, e. g., propositional and first-order logic rules, it is primarily determined by the set of possible truth values of the rules. According to this criterion, extracted rules can be divided into two main groups:

- *Boolean rules*, i. e., formulas of the Boolean logic, such as the propositional *if... then* rules or *M-of-N* rules. As any Boolean formula, they can assume only two different truth values, say *true* and *false*. The tertium-non-datum axiom of the Boolean logic implies that if a Boolean rule has been evaluated and has not been found true, then it automatically must have been found false. That is why methods for the extraction of Boolean rules only need to output rules that, within an apriori set of rules to evaluate, have been found valid in the data.

- *Fuzzy rules*, i. e., formulas of some fuzzy logic, typically formulas of the product logic, Łukasiewicz logic, Gödel logic, or some combination of those three. Their

truth values can be arbitrary elements of some BL-algebra [23]. In the existing methods for the extraction of fuzzy rules, that BL-algebra is always defined on the interval $\langle 0, 1\rangle$.

The proper subject of the present paper are methods for the extraction of fuzzy rules. A closer look at methods of that kind that have been so far proposed in the literature (cf. [4, 9, 15, 20, 34, 39], a survey of many other methods can be found in [35]) reveals that they actually share classification not only with respect to the expressive power, but also with respect to the universality aspect. Indeed, all of them use specialized neural networks, the architectures of which reflect the particular fuzzy logic considered and the syntax of the extracted rules. Needless to say, a network with such properties can hardly be expected to have been constructed and trained in the usually encountered approximation and prediction applications of neural networks, hence it needs to be constructed and trained specifically for the rule extraction task.

An example method based on such a neural network, already implemented and tested with real-world data, will be described in the next section.

3. NEURAL-NETWORKS BASED EXTRACTION OF FUZZY DNF RULES

One of the standard kinds of extracted Boolean rules are rules in the *disjunctive normal form (DNF)*, i. e., rules

$$\Psi \equiv \bigvee_{i=1}^{d} \bigwedge_{j \in C_i} \varphi_{i,j}. \tag{1}$$

in which $C_1, \ldots, C_d$ are non-empty sets, and Ψ, $\varphi_{i,j}$, $i = 1, \ldots, d$, $j \in C_i$, are atomic formulas.

That definition can be in full generality transferred also to a fuzzy logic (for more specific definitions of a fuzzy DNF, based either on the DNF decomposition of Boolean functions, or on fuzzy if-then rules, see [12, 41, 42, 43, 44, 45, 46, 47]) provided the meaning of the connectives $\vee$ and $\wedge$ in (1) is fixed (cf. [22] for the above general definition of a fuzzy DNF in the case of the Gödel logic). In this section, a method for the extraction of such fuzzy DNF rules will be outlined. That method is based on the following four principles:

(i) the underlying neural network computes the truth value of the rule consequent in some model M of the considered fuzzy logic;

(ii) any atomic formulas $\varphi_{i_1,j}$, $\varphi_{i_2,j}$ are interpreted by the same kind of finitely-parametrizable fuzzy sets on a crisp domain D_j (e. g., the value set of some variable);

(iii) the network is trained with a sequence of training pairs $(x^1, y^1), \ldots, (x^t, y^t)$, such that for $k = 1, \ldots, t$, the input component of the kth training pair is a vector $x^k = (x_1^k, \ldots, x_n^k) \in D_1 \times \cdots \times D_n$, and the output component is the desired truth value $y^k \in \langle 0, 1\rangle$ of Ψ provided $\varphi_{i,j}$, $i = 1, \ldots, d$, $j \in C_i$,

are interpreted with respect to x_j^k (i.e., the truth value of $\varphi_{i,j}$ equals the membership of x_j^k in the fuzzy set $\tilde{\varphi}_{i,j}^M$ interpreting $\varphi_{i,j}$ in the model M);

(iv) M is chosen from some considered set of models $\mathcal{M}$ in such a way that the squared Euclidean distance between the computed and the desired outputs of the network, averaged over the training set (mean squared error), is minimal provided each of the atomic formulas $\varphi_{i,j}$ is interpreted by a fuzzy set of a prescribed kind.

Observe that the principle (i) implies that the underlying neural network has 1 output neuron, whereas from (ii) follows that it has n input neurons corresponding to the domains $D_1, \ldots, D_n$ and $|C_1| + \cdots + |C_d|$ hidden neurons corresponding to the considered atomic formulas, where $|C|$ denotes the cardinality of a set C, and $C_i \subset \{1, \ldots, n\}$ for $i = 1, \ldots, d$. For $M \in \mathcal{M}$, let $\| \cdot \|_M$ denote the truth values of formulas of the considered fuzzy logic in the model M. Then the principle (ii) entails a parametrizability of the set $\mathcal{M}$ with a finite number of parameters, and for each $M \in \mathcal{M}$, the principle (iii) implies

$$M = \left(\left(\|\varphi\|_{i,j}^M \right)_{j \in C_i}^{i=1,\ldots,d}, S, T \right) = \left(\left(\tilde{\varphi}_{i,j}^M \right)_{j \in C_i}^{i=1,\ldots,d}, S, T \right), \tag{2}$$

where S and T are, respectively, the s-norm interpreting $\vee$ and the t-norm interpreting $\wedge$. Finally, combining (i) – (ii) with (1) yields

$$\|\Psi\|_M = S_{i=1}^d \left(T_{j \in C_i} \left(\|\varphi_{i,j}\|_M \right) \right) = S_{i=1}^d \left(T_{j \in C_i} \left(\tilde{\varphi}_{i,j}^M \right) \right), \tag{3}$$

whereas the principles (iii) – (iv) in connection with (3) imply that training the network with a sequence of training pairs $(x^1, y^1), \ldots, (x^t, y^t)$ leads to the optimization task

$$M = \arg \min_{M' \in \mathcal{M}} \sum_{k=1}^{t} \left(S_{i=1}^d \left(T_{j \in C_i} \left(\tilde{\varphi}_{i,j}^{M'}(x_j^k) \right) \right) - y^k \right)^2. \tag{4}$$

Due to the finite parametrizability of $\mathcal{M}$, this is a standard task of multidimensional optimization.

For any implementation of the method, the considered set of models $\mathcal{M}$ has to be specified. Taking into account (2) and the principle (ii), this means to specify:

- for each domain D_j, $j = 1, \ldots, n$, the parametrization of the fuzzy sets $\tilde{\varphi}_{i,j}^M$ on D_j, interpreting the atomic formulas $\varphi_{i,j}$, $j \in C_i$, i.e., the number of parameters p_j and the parametrizing mapping $\pi_j : \Re^{p_j} \to \mathcal{F}(D_j)$, where $\mathcal{F}(D_j)$ denotes the set of fuzzy sets on D_j and π_j fulfils $(\exists\, a_{i,j}^M \in \Re^{p_j})\ \tilde{\varphi}_{i,j}^M = \pi_j(a_{i,j}^M)$, for $i = 1, \ldots, d$, $j \in C_i$, $M \in \mathcal{M}$;
- the particular fuzzy logic considered, which in turn determines the considered s-norm S and the t-norm T.

In the implementation of the method at the Institute of Computer Science in Prague, the user can combine any of the parametrizations in table on page 302 with either the Łukasiewicz logic [23], entailing the t-norm

$$T_{\mathrm{L}}(x, y) = \max(x + y - 1, 0) \,|\, x, y \in \langle 0, 1 \rangle, \tag{5}$$

and s-norm

$$S_{\mathrm{L}}(x,y) = \min(x+y,1) \,|\, x,y \in \langle 0,1\rangle, \tag{6}$$

or the product-Łukasiewicz logic [29] with the t-norm

$$T_{\mathrm{PL}}(x,y) = xy \,|\, x,y \in \langle 0,1\rangle, \tag{7}$$

and s-norm

$$S_{\mathrm{PL}}(x,y) = x+y-xy \,|\, x,y \in \langle 0,1\rangle. \tag{8}$$

At the present implementation, $D_j = \Re$ is assumed for all $j = 1,\ldots,n$, and only one parametrization at a time can be chosen, which is then used for the interpretation of all $\varphi_{i,j}$, $i = 1,\ldots,d$, $j \in C_i$.

parametrization	number of parameters	parametrizing mapping
Gaussian	2	$\pi(a,b) = \Gamma_{(a,b)} \in \mathcal{F}(\Re) \mid a \in \Re, b > 0,$ where $(\forall x \in \Re)\ \Gamma_{(a,b)}(x) = e^{-\frac{(x-a)^2}{2b}}$
symmetric triangular	2	$\pi(a,b) = \Delta^s_{(a,b)} \in \mathcal{F}(\Re) \mid a \in \Re, b > 0,$ where $(\forall x \in \Re)\ \Delta^s_{(a,b)}(x) = \max(0, \frac{1-\lvert x-a\rvert}{b})$
triangular	3	$\pi(a,b,c) = \Delta_{(a,b,c)} \in \mathcal{F}(\Re) \mid a < b < c,$ where $(\forall x \in \Re)\ \Delta_{(a,b,c)}(x) = \max(0, \min(\frac{x-a}{b-a}, \frac{c-x}{c-b}))$;
bell-shaped	3	$\pi(a,b,c) = B_{(a,b,c)} \in \mathcal{F}(\Re) \mid a,b,c \in \Re, a \neq 0,$ where $(\forall x \in \Re)\ B_{(a,b,c)}(x) = \frac{1}{1+\left(\frac{x-c}{a}\right)^{2b}},$
sigmoidal spline	2	$\pi(a,b) = S^{\varsigma}_{(a,b)} \in \mathcal{F}(\Re) \mid a < b,$ where $S^{\varsigma}_{(a,b)}(x) = \begin{cases} 0 \mid x \leq a; \\ 2\left(\frac{x-a}{b-a}\right)^2 \mid a \leq x \leq \frac{a+b}{2} \\ 1-2\left(\frac{b-x}{b-a}\right)^2 \mid \frac{a+b}{2} \leq x \leq b \\ 1 \mid x \geq b, \end{cases}$
sigmoidal	2	$\pi(a,b) = S_{(a,b)} \in \mathcal{F}(\Re) \mid a \in \Re, b > 0,$ where $(\forall x \in \Re)\ S_{(a,b)}(x) = \frac{1}{1+e^{-b(x-a)}}$
decreasing spline	2	$\pi(a,b) = D_{(a,b)} \in \mathcal{F}(\Re) \mid a < b,$ where $D_{(a,b)}(x) = \begin{cases} 1 \mid x \leq a; \\ 1-2\left(\frac{x-a}{b-a}\right)^2 \mid a \leq x \leq \frac{a+b}{2} \\ 2\left(\frac{b-x}{b-a}\right)^2 \mid \frac{a+b}{2} \leq x \leq b \\ 0 \mid x \geq b, \end{cases}$

For the resulting combinations of parametrization with t-norms and s-norms, (2) and (4) yield the final formulation of the optimization task to be solved, for example:

$$M_{\Gamma,\mathrm{L}} = \arg\min_{(a_{i,j}\in\Re,b_{i,j}>0)_{j\in C_i}^{i=1,\ldots,d}} \tag{9}$$

$$\sum_{k=1}^{t}\left(\min\left(\sum_{i=1}^{d}\left(\max\left(\sum_{j\in C_i} e^{-\frac{(x_j^k-a_{i,j})^2}{2b_{i,j}}} - |C_i|+1,0\right),1\right)-y^k\right)^2,$$

$$M_{\Gamma,\mathrm{PL}} = \arg\min_{(a_{i,j}\in\Re,b_{i,j}>0)_{j\in C_i}^{i=1,\ldots,d}} \tag{10}$$

$$\sum_{k=1}^{t}\left(1-\prod_{i=1}^{d}(1-e^{-\sum_{j\in C_i}\frac{(x_j^k-a_{i,j})^2}{2b_{i,j}}})-y^k\right)^2,$$

$$M_{\Delta^s,\mathrm{L}} = \arg\min_{(a_{i,j}\in\Re,b_{i,j}>0)_{j\in C_i}^{i=1,\ldots,d}} \tag{11}$$

$$\sum_{k=1}^{t}\left(\min\left(\sum_{i=1}^{d}\max\left(\sum_{j\in C_i}\max\left(0,\frac{1-|x_j^k-a_{i,j}|}{b_{i,j}}\right)-|C_i|+1,0\right),1\right)-y^k\right)^2,$$

$$M_{\Delta^s,\mathrm{PL}} = \arg\min_{(a_{i,j}\in\Re,b_{i,j}>0)_{j\in C_i}^{i=1,\ldots,d}} \tag{12}$$

$$\sum_{k=1}^{t}\left(1-\prod_{i=1}^{d}\left(1-\prod_{j\in C_i}\max\left(0,\frac{1-|x_j^k-a_{i,j}|}{b_{i,j}}\right)-y^k\right)\right)^2,$$

$$M_{\mathrm{B},\mathrm{L}} = \arg\min_{(a_{i,j},b_{i,j},c_{i,j}\in\Re,a_{i,j}\neq 0)_{j\in C_i}^{i=1,\ldots,d}} \tag{13}$$

$$\sum_{k=1}^{t}\left(\min\left(\sum_{i=1}^{d}\max\left(\sum_{j\in C_i}\frac{1}{1+\left(\frac{x_j^k-c_{i,j}}{a_{i,j}}\right)^{2b_{i,j}}}-|C_i|+1,0\right),1\right)-y^k\right)^2,$$

$$M_{\mathrm{B},\mathrm{PL}} = \arg\min_{(a_{i,j},b_{i,j},c_{i,j}\in\Re,a_{i,j}\neq 0)_{j\in C_i}^{i=1,\ldots,d}} \tag{14}$$

$$\sum_{k=1}^{t}\left(1-\prod_{i=1}^{d}\left(1-\prod_{j\in C_i}\frac{1}{1+\left(\frac{x_j^k-c_{i,j}}{a_{i,j}}\right)^{2b_{i,j}}}\right)-y^k\right)^2.$$

For illustration, Figures 2 and 3 show the same 2-dimensional cut of the truth values of $\|\Psi\|_M$ for models M fulfilling (11) and (14), respectively. The neural networks in these figures have been trained with data from a recent application of ANNs to material science [27, 28].

Though the implementation covers the most important special cases, the method itself is actually applicable to a much broader class of problems, due to the generality of the set of models $\mathcal{M}$. Indeed, $\mathcal{M}$ is only required to comply with (2), otherwise it can be quite arbitrary, in particular it can impose arbitrary additional restrictions to the included models. Such restrictions will now be briefly illustrated on an example from a currently starting application of the method to the results of an EEG spectral analysis in neurophysiology [19].

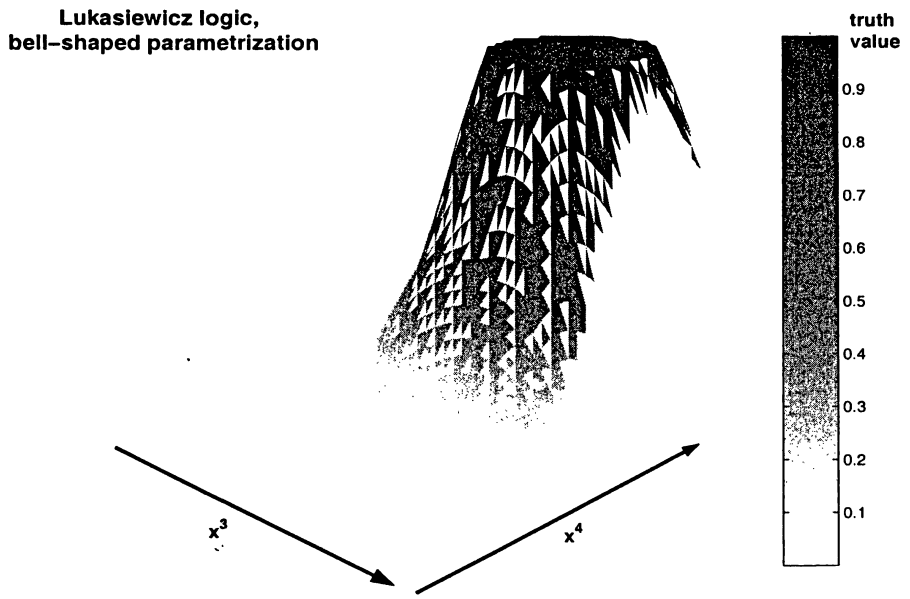

Fig. 2. A 2-dimensional cut corresponding to the dimensions x_3 and x_4 of the truth value $\|\Psi\|_M = 1 - \prod_{i=1}^{10}\left(1 - e^{-\sum_{j=1}^{13}\frac{(x_j - a_{i,j})^2}{2b_{i,j}}}\right)$, where $a_{i,j}$ and $b_{i,j}$ for $i = 1,\ldots,10$, $j = 1,\ldots,13$ are obtained from (11), $x_8 = 1 - x_3 - x_4$, and $x_j = 0$ for j =1,2,5–7,9–13.

The data from an EEG spectral analysis contain amplitudes of EEG signal components for all included frequencies of the spectrum. However, knowledge about EEG spectra is usually formulated in terms of differently looking bands of the spectrum (δ-, θ-, α- and β-bands). This has the following consequences for the fuzzy sets $(\tilde{\varphi}_{i,j}^M)_{i=1,\ldots,d,j\in C_i}$ in (2):

(i) $D_j = \Re$ for $j = 1,\ldots,n$;

(ii) $\tilde{\varphi}_{i,j}^M = S(\tilde{\delta}_i^M, \tilde{\theta}_i^M, \tilde{\alpha}_i^M, \tilde{\beta}_i^M)$ for $i = 1,\ldots,d$, $j \in C_i = \{1,\ldots,n\}$, where $\tilde{\delta}_i^M, \tilde{\theta}_i^M, \tilde{\alpha}_i^M, \tilde{\beta}_i^M$ are fuzzy sets describing the respective band of the spectrum;

(iii) there exist a number $p \in \mathcal{N}$, a parametrizing mapping $\pi : \Re^p \to \mathcal{F}(\Re)$, and parameters $a_{i,\delta}^M, a_{i,\theta}^M, a_{i,\alpha}^M, a_{i,\beta}^M \in \Re^p$) such that $\tilde{\xi}_i^M = \pi(a_{i,\xi})$ for $\xi \in \{\delta,\theta,\alpha,\beta\}$, $i = 1,\ldots,d$, $M \in \mathcal{M}$;

(iv) there exist numbers $L_\delta, L_\theta, L_\alpha, L_\beta, U_\delta, U_\theta, U_\alpha, U_\beta \in \Re$ such that $L_\xi < U_\xi$ and $\tilde{\xi}_i^M \,|\, (-\infty, L_\xi\rangle = \tilde{\xi}_i^M \,|\, \langle U_\xi, \infty) = 0$ for $\xi \in \{\delta,\theta,\alpha,\beta\}$, $i = 1,\ldots,d$, $M \in \mathcal{M}$.

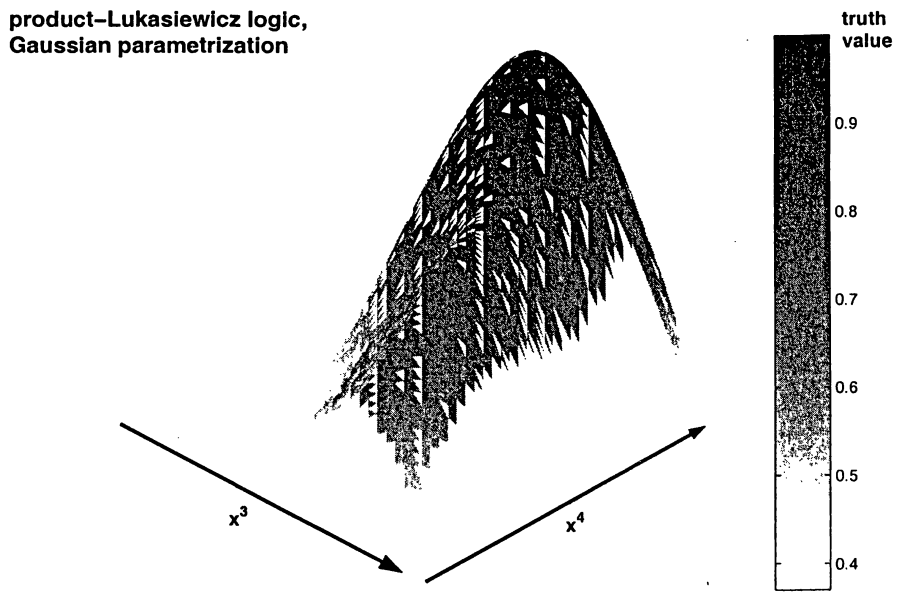

Fig. 3. A 2-dimensional cut corresponding to the dimensions x_3 and x_4 of the truth value $\|\Psi\|_M = \min\left(\sum_{i=1}^{5} \max\left(\sum_{j=1}^{13} \frac{1}{1+\left(\frac{x_j^k - c_{i,j}}{a_{i,j}}\right)^{2b_{i,j}}} - 12, 0\right), 1\right)$, where $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$ for $i = 1, \ldots, 10$, $j = 1, \ldots, 13$ are obtained from (14), $x_8 = 1 - x_3 - x_4$, and $x_j = 0$ for j =1,2,5–7,9–13.

Hence, the set of models $\mathcal{M}$ gets restricted to

$$\mathcal{M} = \Big\{ M = ((\tilde{\varphi}_{i,j}^M)_{j \in C_i}^{i=1,\ldots,d}, S, T) : (\forall i \in \{1, \ldots, d\})(\exists\, \tilde{\delta}_i^M, \tilde{\theta}_i^M, \tilde{\alpha}_i^M, \tilde{\beta}_i^M \in \mathcal{F}(\Re))$$
$$(\forall \xi \in \{\delta, \theta, \alpha, \beta\})(\exists\, a_{i,\xi}^M \in \Re^p)\ \tilde{\xi}_i^M = \pi(a_{i,\xi})\ \&\ \tilde{\xi}_i^M \mid (-\infty, L_\xi\rangle = \tilde{\xi}_i^M \mid \langle U_\xi, \infty) = 0$$
$$\&(\forall j \in \{1, \ldots, n\})\ \tilde{\varphi}_{i,j}^M = S(\tilde{\delta}_i^M, \tilde{\theta}_i^M, \tilde{\alpha}_i^M, \tilde{\beta}_i^M) \Big\}. \tag{15}$$

4. IS EXTRACTION FROM GENERAL MULTILAYER PERCEPTRONS POSSIBLE?

As was mentioned in Section 2, all existing methods for ANN-based extraction of fuzzy logic rules rely on some highly specialized neural networks, thus they are hardly portable to networks commonly encountered in applications. Recent results by Amato, Porto, Aguzzoli and Mundici on the connection between fuzzy logic and piecewise-linear functions $\langle 0, 1\rangle^n \to \langle 0, 1\rangle$ with rational coefficients [2, 5] indicate that this may not need to be the case. Both quoted papers show that such piecewise-linear functions (i. e., rational generalizations of McNaughton functions) are actually

fuzzy sets on $\langle 0,1\rangle^n$ interpreting particular formulas of some fuzzy logic, or in other words, they are fuzzy logic functions represented by formulas of the respective logic. In [2], that logic is the logic $\exists$Ł, a fragment of the infinite-valued Łukasiewicz predicate logic, whereas in [5], it is the Esteva–Godo–Montagna logic Ł$\Pi\frac{1}{2}$ [18]. Those results can be considered a direct generalization of the classical McNaughton theorem [33]. Though none of both results concerns neural networks, they nevertheless imply a principal possibility to extract fuzzy logic rules from multilayer perceptrons with continuous activation functions, the kind of ANNs that is most commonly used in applications. Indeed:

1. Due to the density of the set Q of rational numbers within the set $\Re$ of real numbers, any piecewise-linear function $\langle 0,1\rangle^n \to \langle 0,1\rangle$ is arbitrarily close (in the metrics on $\mathcal{C}(\langle 0,1\rangle^n)$) to a piecewise-linear function $\langle 0,1\rangle^n \to \langle 0,1\rangle$ with rational coefficients, i. e., to the interpretation of some formula of either the logic Ł$\Pi\frac{1}{2}$, or the logic $\exists$Ł.

2. Any function computed by a multilayer perceptron with the input space $\langle 0,1\rangle^n$ and continuous activation functions is arbitrarily close to a function computed by a multilayer perceptron with piecewise-linear activation functions, which in turn is a piecewise-linear function $\langle 0,1\rangle^n \to \langle 0,1\rangle$ [26, 32].

3. An alternative argument to 1–2 is the fact that any continuous mapping $\langle 0,1\rangle^n \to \langle 0,1\rangle$ (in particular, any continuous mapping $\langle 0,1\rangle^n \to \langle 0,1\rangle$ computed by a multilayer perceptron with the input space $\langle 0,1\rangle^n$) is arbitrarily close to a function represented by a fuzzy normal form of a specific kind [41]. However, such normal forms require the language of the Łukasiewicz logic to be extended with truth constants for all numbers from $\langle 0,1\rangle$ (or at least for all rationals from $\langle 0,1\rangle$), which is not the case for the fuzzy logics considered in [2] and [5].

Moreover, both quoted papers provide a constructive proof of the representability of piecewise-linear functions $\langle 0,1\rangle^n \to \langle 0,1\rangle$ with rational coefficients by formulas of the considered fuzzy logic, a proof in both cases heavily relying on Mundici's constructive proof of the McNaughton theorem [10, 37] (which is different from the more recent constructive proof by Perfilieva and Tonis [40, 41]). Those constructive proofs, together with an algorithm for the approximation according to (ii) above [26], already allow to formulate algorithms for the extraction of formulas of the respective fuzzy logic from a multilayer perceptron with continuous activation functions (for the approximation according to (i), no algorithm is needed because all computations are always performed with rational numbers). Here, the main steps of an algorithm relying on [2] are sketched. To facilitate the formulation of the algorithm and the subsequent discussion, several simplifying assumptions will be adopted:

- the considered multilayer perceptron has only one output neuron (an extension to perceptrons with more output neurons is possible through splitting the original network into several perceptrons sharing the input and hidden neurons and the connections between them, and through applying the algorithm below to each of those perceptrons);

- the considered multilayer perceptron has only one hidden layer (an extension to perceptrons with more hidden layers is possible through repeating the steps 1 – 3 for each of them, proceeding from the last hidden layer to the first hidden layer);
- the somatic operation at the output neuron doesn't include the activation function (an extension to output neurons that include the activation function is possible through starting the algorithm with steps 1 – 3 applied to the output neuron);
- for the purpose of rule extraction, the input space of the considered multilayer perceptron is restricted from a whole Euclidean space $\Re^n$ to the unit cube $\langle 0,1\rangle^n$ (this can be in a standard way extended to all closed cubes, and further to all compact sets in $\Re^n$, which are sufficient for real-world applications of multilayer perceptrons).

On those assumptions, the algorithm can be formulated as follows:

Input: A required precision $\varepsilon > 0$ for the approximation according to 2. above, a multilayer perceptron with one hidden layer, n_I input neurons, n_H hidden neurons and an activation function f, and a function F computed by that perceptron and defined

$$(\forall x \in \Re)\ F(x) = \sum_{h=1}^{n_H} w_h f\left(\sum_{i=1}^{n_I} w_{h,i} x_i + b_h\right) + b_O, \tag{16}$$

where $W_{IH} = (w_{h,i})_{i=1,\ldots,n_I}^{h=1,\ldots,n_H} \neq 0$ is a matrix of the weights of connections between input and hidden neurons, $b_H = (b_1, \ldots, b_{n_H})$ is a vector of the biases of hidden neurons, $W_{HO} = (w_1, \ldots, w_{n_H}) \neq 0$ is a vector of the weights of connections between hidden neurons and the output neuron, and b_O is a bias of the output neuron.

Step 1. Let $U = \max\limits_{h=1,\ldots,n_H} \max\limits_{u\in\langle 0,1\rangle^n} |(w_{h,1}, \ldots, w_{h,n_I})'u|$, m be the smallest integer such that $|f(u) - f(u')| < \frac{\varepsilon}{\sum_{h=1}^{n_H} |w_h|}$ whenever $|u - u'| < \frac{2U}{m}$, and define a piecewise-linear function $g : \langle -U, U\rangle \to \langle 0,1\rangle$ by

$$(\forall u \in \langle -U,U\rangle)\ g(u) = f\left(-U + \left[\frac{m(u+U)}{2U}\right]\frac{2U}{m}\right) + \left(\frac{m(u+U)}{2U} - \left[\frac{m(u+U)}{2U}\right]\right)$$
$$\left(f\left(-U + \left(\left[\frac{m(u+U)}{2U}\right]+1\right)\frac{2U}{m}\right) - f\left(-U + \left[\frac{m(u+U)}{2U}\right]\frac{2U}{m}\right)\right), \tag{17}$$

where $[u]$ denotes the integer part of a real number u.

Step 2. Create a polyhedral complex $\mathcal{P}_{n_I}$ partitioning $\langle 0,1\rangle^{n_I}$:

$$\mathcal{P}_{n_I} = \left\{P \subset \langle 0,1\rangle^{n_I} : W_{IH}(P) = \left(\left\langle -U - b_1 + (i_1 - 1)\frac{2U}{m}, -U - b_1 + i_1\frac{2U}{m}\right\rangle \times \ldots\right.\right. \tag{18}$$
$$\left.\left. \cdots \times \left\langle -U - b_{n_H} + (i_{n_H} - 1)\frac{2U}{m}, -U - b_{n_H} + i_{n_H}\frac{2U}{m}\right\rangle,\ i_1, \ldots, i_{n_H} = 1, \ldots, m\right\},$$

where the notation $W_{IH}(P)$ for $\{W_{IH}(x) : x \in P\}$ is used.

Step 3. Define a piecewise-linear mapping $G : \langle 0,1\rangle^{n_I} \to \langle 0,1\rangle$ by

$$(\forall x \in \langle 0,1\rangle^{n_I})\ G(x) = \sum_{h=1}^{n_H} w_h g\left(\sum_{i=1}^{n_I} w_{h,i} x_i + b_h\right) + b_O, \tag{19}$$

so that, due to Steps 1–2, G approximates F at $\langle 0,1\rangle^{n_I}$ within the precision ε.

Step 4. Triangularize the polyhedral complex $\mathcal{P}_{n_I}$ into a simplicial complex $\mathcal{S}_{n_I}$ (see, e. g., [17] for details).

Step 5. Create a polyhedral complex $\mathcal{P}_{n_I+1}$ partitioning $\langle 0,1\rangle^{n_I+1}$:

$$\begin{aligned}\mathcal{P}_{n_I+1} = \big\{P \subset \langle 0,1\rangle^{n_I+1} : (\exists S \in \mathcal{S}_{n_I})\ P \text{ is a polyhedron, and has either}\\ \text{the vertices } (s_1,0),\dots,(s_{n_I+1},0),(s_1,f(s_1)),\dots,(s_{n_I+1},f(s_{n_I+1})),\\ \text{or the vertices} (s_1,f(s_1)),\dots,(s_{n_I+1},f(s_{n_I+1})),(s_1,1),\dots,(s_{n_I+1},1),\\ \text{where } s_1,\dots,s_{n_I+1} \text{ are the vertices of } S\big\}.\end{aligned} \tag{20}$$

Step 6. Triangularize the polyhedral complex $\mathcal{P}_{n_I+1}$ into a simplicial complex $\mathcal{S}_{n_I+1}$.

Step 7. As long as there exists an (n_I+1)-dimensional simplex $S \in \mathcal{S}_{n_I+1}$ such that $\det \mathcal{I}_S \neq \pm 1$ for its matrix $\mathcal{I}_S$ in homogeneous integer coordinates, refine $\mathcal{S}_{n_I+1}$ through adding such a vertex s inside of S that for any subsimplex S' of the resulting simplicial partitioning of S, $\det \mathcal{I}'_S < \det \mathcal{I}_S$, thus finally arriving to a unimodular refinement $\mathcal{U}$ of $\mathcal{S}_{n_I+1}$ (see [10] for details).

Step 8. For each vertex $v \in V$, where $V = \big\{v = \big(\frac{a_1^v}{d},\dots,\frac{a_{n_I+1}^v}{d}\big) \in Q^{n_I+1} : (\exists U \in \mathcal{U})\ v$ is a vertex of U & $a_1^v,\dots,a_{n_I+1}^v, d$ are integers & $d > 0$ & $\frac{a_{n_I+1}^v}{d} \leq G\big(\frac{a_1^v}{d},\dots,\frac{a_{n_I}^v}{d}\big)\big\}$, construct a Schauder hat of v with respect to $\mathcal{U}$, i. e., a piecewise-linear function with integer coefficients $H_{\mathcal{U}}^v : \langle 0,1\rangle^{n_I+1} \to \langle 0,1\rangle$ that is linear over each $U \in \mathcal{U}$ and fulfills $H_{\mathcal{U}}^v(v) = \frac{1}{d}$, $H_{\mathcal{U}}^v(v') = 0$ for all other vertices v' of any $U \in \mathcal{U}$ (see [10] for the existence of such a function, and for details of that construction).

Step 9. Define a formula $(\exists X_{n_I+1})\ \Phi(X_1,\dots,X_{n_I+1})$ of the logic $\exists$ Ł, in which the formula Φ with n_I+1 free variables $X_1,\dots,X_{n_I+1}$ is defined gradually starting with formulas that represent individual linear pieces of individual Schauder hats, and proceeding via formulas that represent entire Schauder hats, in such a way that the final formula $(\exists X_{n_I+1}\ \Phi(X_1,\dots,X_{n_I+1})$ represents $\max_{x_{n_I+1}\in\langle 0,1\rangle} \sum_{v\in V} a_{n_I+1}^v H_{\mathcal{U}}^v$ (see [10] for details).

Output: The formula $(\exists X_{n_I+1})\ \Phi(X_1,\dots,X_{n_I+1})$ defined in Step 8; due to the equality

$$(\forall x=(x_1,\ldots,x_{n_I})\in\langle 0,1\rangle^{n_I})\ \max_{x_{n_I+1}\in\langle 0,1\rangle}\sum_{v\in V} a^v_{n_I+1}H^v_{\mathcal{U}}(x_1,\ldots,x_{n_I},x_{n_I+1})=G(x), \tag{21}$$

proven in [2], this formula represents also G.

Already a first look at this algorithm reveals that, even when neglecting assignments, concatenations and comparisons, most steps involve computations of exponential complexity, and the last step (Step 9) even computations of doubly-exponential complexity:

(i) In Step 2, a polyhedral complex $\mathcal{P}_{n_I}$ with $|\mathcal{P}_{n_I}| = O(m^{n_H})$ is created through solving m^{n_H} linear equations $W_{IH}(x) = \left(i_1\frac{2U}{m},\ldots,i_{n_H}\frac{2U}{m}\right)$ for $i_1,\ldots,i_{n_H} = 0,\ldots,m$.

(ii) In Step 4, each polyhedron from $\mathcal{P}_{n_I}$ is triangularized into $O\left(2^{\frac{n_H^2}{2}}\right)$ simplices (see, e. g., [7]), thus $\mathcal{P}_{n_I}$ is altogether triangularized into $|\mathcal{S}_{n_I}| = O\left(2^{\frac{n_H^2}{2}}m^{n_H}\right)$ simplices.

(iii) In Step 5, a polyhedral complex $\mathcal{P}_{n_I}$ with $|\mathcal{P}_{n_I+1}| = O\left(2^{\frac{n_H^2}{2}}m^{n_H}\right)$ polyhedra is created through constructing 2 polyhedra over each simplex $S\in\mathcal{S}_{n_I}$.

(iv) In Step 6, each polyhedron from $\mathcal{P}_{n_I+1}$ is triangularized into $O\left(n_H^{\frac{n_H}{2}}\right)$ simplices, thus $\mathcal{P}_{n_I+1}$ is altogether triangularized into $|\mathcal{S}_{n_I+1}| = O\left(2^{\frac{n_H^2}{2}}m^{n_H}n_H^{\frac{n_H}{2}}\right)$ simplices.

(v) In any of the repeatedly performed iterations of Step 7, a particular simplex S from the current refinement $\mathcal{S}$ of the simplicial complex $\mathcal{S}_{n_I+1}$ is splitted into $O(n_H)$ simplices such that for each of them, $\det\mathcal{I}'_S < \det\mathcal{I}_S$, thus as many as $|\mathcal{S}_{n_I+1}|$ iterations may be needed to decrease the current value of $D_{\mathcal{S}} = \max_{S\in\mathcal{S}}|\det\mathcal{I}_S|$ from $D_{\mathcal{S}_{n_I+1}}$ to $D_{\mathcal{S}_{n_I+1}}-1$, and $\mathcal{S}_{n_I+1}$ is splitted into $O\left(n_H|\mathcal{S}_{n_I+1}|\right)$ simplices during that time; repeating this until the unimodular simplicial complex $\mathcal{U}$ with $D_{\mathcal{U}} = 1$ is reached implies $|\mathcal{U}| = O\left(n_H^{D_{\mathcal{S}_{n_I+1}}}|\mathcal{S}_{n_I+1}|\right) = O\left(2^{\frac{n_H^2}{2}}m^{n_H}n_H^{\left(\frac{n_H}{2}+D_{\mathcal{S}_{n_I+1}}\right)}\right)$.

(vi) In Step 8, for each of the $O(|\mathcal{U}|)$ vertices $v\in V$, an (n_H+2)-dimensional system of linear equations has to be solved to construct each of the $\ell^v_{\mathcal{U}} \le |\mathcal{U}|$ nonzero linear pieces of the Schauder hat $H^v_{\mathcal{U}}$, thus altogether $O(|\mathcal{U}|^2) = O\left(2^{n_H^2}m^{2n_H}n_H^{(n_H+2D_{\mathcal{S}_{n_I+1}})}\right)$ such systems of linear equations have to be solved.

(vii) In Step 9, for each of the $O(|\mathcal{U}|)$ vertices $v\in V$ and each permutation σ of the number $\ell^v_{\mathcal{U}}$ of different linear pieces $g_1,\ldots,g_{\ell^v_{\mathcal{U}}}$ of the Schauder hat $H^v_{\mathcal{U}}$, it has to be decided whether the polyhedron $P_\sigma = \{x\in\Re^{n_I+1} : g_{\sigma(1)}(x)\ge\cdots\ge g_{\sigma(\ell^v_{\mathcal{U}})}(x)\}$ is (n_I+1)-dimensional, through checking the regularity of the system of vertices $P_\sigma\cap V$, thus altogether the regularity of $O(\ell^v_{\mathcal{U}}!) = O\left(e^{O\left(2^{\frac{n_H^2}{2}}m^{n_H}n_H^{\left(\frac{n_H}{2}+D_{\mathcal{S}_{n_I+1}}\right)}\right)}\right)$ systems of $O\left(2^{\frac{n_H^2}{2}}m^{n_H}n_H^{\left(\frac{n_H}{2}+D_{\mathcal{S}_{n_I+1}}\right)}\right)$ vectors from $\Re^{n_I+1}$ has to be checked.

Observe that the highest complexity is connected with the construction of formulas corresponding to Schauder hats. Since that construction has been taken over from Mundici's constructive proof of the McNaughton theorem [37, 10], the overall doubly-exponential complexity pertains already to the representation of piecewise-linear functions with integer coefficients by formulas of Lukasiewicz propositional logic according to that proof, and is not specific to the presented algorithm for the representation of piecewise-linear functions with rational coefficients by formulas of the logic $\exists$Ł. Moreover, it can be shown that also the alternative constructive proof of the McNaughton theorem in [40, 41] finally leads to a doubly-exponential complexity. Due to the complexity of involved computations, the above algorithm can be viewed merely as a demonstration that the extraction of formulas of fuzzy logic from general multilayer perceptrons is principally possible. However, further research is needed to arrive to an algorithm that will be practically applicable.

5. CONCLUSION

The paper surveyed the task of extracting fuzzy logic rules from data by means of artificial neural networks. It described a particular method of that kind, based on a fuzzy generalization of DNF rules, which had already been successfully employed in several practical applications. On the other hand, it also demonstrated that the common feature of all existing fuzzy rule extraction methods to rely on highly specialized networks constructed directly for the rule extraction task is not a principal necessity. An algorithm has been proposed for the extraction of formulas of the infinite-valued Lukasiewicz logic from neural networks as general as multilayer perceptrons with continuous activation functions. That generality makes the algorithm theoretically attractive for the extraction of fuzzy rules from numerous trained multilayer perceptrons that are available in real-world applications. However, its applicability to this end is hindered by its high computational complexity.

Simplifications of the algorithm to decrease its complexity and make it feasible for practical applications are the topic of ongoing research. That research is driven by the following ideas:

- To restrict, in Steps 2 and 4, the cardinalities of the involved polyhedral and simplicial complexes through considering only those polyhedra and simplices that contain the input component x^k of at least one training pair (x^k, y^k). In that way, the cardinalities of $|\mathcal{P}_{n_I}|$ and $|\mathcal{S}_{n_I}|$ get restricted to $O(t)$, compared to $O(m^{n_H})$ and $O\big(2^{\frac{n_H^2}{2}} m^{n_H}\big)$, respectively.
- To seek alternatives to Steps 5 – 9. In this respect, the alternative constructive proof of the McNaughton theorem in [40, 41] is very inspiring, but since it covers only Steps 7 – 9, at least an alternative to Steps 5 – 6 needs to be sought anyway.

Most important for the feasibility of any proposed simplification of the algorithm above will be results of its testing on real-world problems and comparison with the method for extracting fuzzy DNF rules from Section 3 and / or with other methods based on specialized networks. That comparison needs to include not only the final

computational complexity of the proposed simplification, compared to the complexity of retraining the specialized network, but also the quality of the extracted rules, especially their comprehensibility and accuracy.

Due to the approximations involved in extending the results [2, 5] from rational McNaughton functions to continuous functions computed by multilayer perceptrons (see 1–2 in Section 4), the interpretation of the extracted formula actually only approximates that continuous function, except for the case when the computed function is a rational McNaughton function. Needless to say, simplifications of the algorithm cannot improve this situation. On the contrary, they can lead to approximations even for rational McNaughton functions. It is interesting to compare the resulting approximations with several methods for approximation of continuous functions based on specific kinds of fuzzy normal forms [41, 47]. In those methods, the approximation is in fact based on crisp sets (on sufficiently small balls from the domain of the approximated function, or on products of sufficiently small balls from its domain and range, covering the function). Consequently, the approximation can be actually described also by means of Boolean logic in those methods. In contrast, the method outlined in Section 4 is based on proper fuzzy sets, represented by formulas of Łukasiewicz logic, and the resulting approximation could not be described by means of Boolean logic.

ACKNOWLEDGEMENT

The research reported in this paper has been supported by the grant No. A1030004, "Mathematical Foundations of Inference Under Vagueness and Uncertainty", of the Grant Agency of Academy of Sciences of the Czech Republic. The author is indebted to Přemysl Posledník for implementing a substantial part of the method presented in Section 3.

(Received April 23, 2004.)

REFERENCES

[1] J. M. Adamo: Data Mining for Association Rules and Sequential Patterns: Sequential and Parallel Algorithms. Springer–Verlag, Berlin 2001.

[2] S. Aguzzoli and D. Mundici: Weierstrass approximations by Łukasiewicz formulas with one quantified variable. In: 31st IEEE Internat. Symposium on Multiple-Valued Logic, 2001.

[3] J. A. Alexander and M. C. Mozer: Template-based procedures for neural network interpretation. Neural Networks *12* (1999), 479–498.

[4] P. Amato, A. Di Nola, and B. Gerla: Neural networks and rational Łukasiewicz logic. J. Multiple-Valued Logic and Soft Computing (accepted for publication).

[5] P. Amato and M. Porto: An algorithm for the automatic generation of logical formula representing a control law. Neural Network World *10* (2000), 777–786.

[6] R. Andrews, J. Diederich, and A. B. Tickle: Survey and critique of techniques for extracting rules from trained artificical neural networks. Knowledge-based Systems *8* (1995), 378–389.

[7] M. Bern, L. P. Chew, D. Eppstein, and J. Ruppert: Dihedral bounds for mesh generation in high dimensions. In: Proc. Sixth ACM-SIAM Symposium on Discrete Algorithms, ACM, San Francisco 1995, pp. 189–196.

[8] M. Berthold and D. Hand, editors: Intelligent Data Analysis. An Introduction. Springer–Verlag, Berlin 1999.
[9] J. Chen and J. Liu: Using mixture principal component analysis networks to extract fuzzy rules from data. Indust. Engrg. Chemistry Research *39* (2000), 2355–2367.
[10] L. O. Cignoli, I. M. L. D'Ottaviano, and D. Mundici: Algebraic Foundations of Many-valued Reasoning. Kluwer Academic Publishers, Dordrecht 2000.
[11] A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay: Symbolic knowledge extraction from artificial neural networks: A sound approach. Artificial Intelligence *125* (2001), 155–207.
[12] M. Daňková and I. Perfilieva: Logical approximation II. Soft Computing *7* (2003), 228–233.
[13] L. De Raedt: Interactive Theory Revision: An Inductive Logic Programming Approach. Academic Press, London 1992.
[14] W. Duch, R. Adamczak, and K. Grabczewski: Extraction of logical rules from neural networks. Neural Processing Lett. *7* (1998), 211–219.
[15] W. Duch, R. Adamczak, and K. Grabczewski: A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Trans. Neural Networks *11* (2000), 277–306.
[16] S. Dzeroski and N. Lavrac: Relational Data Mining. Springer–Verlag, Berlin 2001.
[17] H. Edelsbrunner: Algorithms in Combinatorial Geometry. Springer–Verlag, Heidelberg 1987.
[18] F. Esteva, L. Godo, and F. Montagna: The ŁΠ and$\text{ŁΠ}\frac{1}{2}$ logics: Two complete fuzzy systems joiningŁukasiewicz and product logic. Arch. Math. Logic *40* (2001), 39–67.
[19] J. Faber, M. Novák, P. Svoboda, and V. Tatarinov: Electrical brain wave analysis during hypnagogium. Neural Network World *13* (2003), 41–54.
[20] G. D. Finn: Learning fuzzy rules from data. Neural Computing Appl. *8* (1999), 9–24.
[21] A. A. Freitas: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer–Verlag, Berlin 2002.
[22] M. Gehrke, C. L. Walker, and E. A. Walker: Normal forms and truth tables for fuzzy logics. Fuzzy Sets and Systems *138* (2003), 25–51.
[23] P. Hájek: Metamathematics of Fuzzy Logic. Kluwer Academic Publishers, Dordrecht 1998.
[24] P. Hájek and T. Havránek: Mechanizing Hypothesis Formation. Springer–Verlag, Berlin 1978.
[25] M. J. Healy and T. P. Caudell: Acquiring rule sets as a product of learning in a logical neural architecture. IEEE Trans. Neural Networks *8* (1997), 461–474.
[26] M. Holeňa: Extraction of logical rules from data by means of piecewise-linear neural networks. In: Proc. 5th Internat. Conference on Discovery Science, Springer–Verlag, Berlin 2002, pp. 192–205.
[27] M. Holeňa and M. Baerns: Artificial neural networks in catalyst development. In: Experimental Design for Combinatorial and High Throughput Materials Development (J. N. Cawse, ed.), Wiley, Hoboken 2003, pp. 163–202.
[28] M. Holeňa and M. Baerns: Feedforward neural networks in catalysis. A tool for the approximation of the dependency of yield on catalyst composition, and for knowledge extraction. Catalysis Today *81* (2003), 485–494.
[29] R. Horčík and P. Cintula: Product Łukasiewicz logic. Arch. Math. Logic *43* (2004), 477–503.
[30] M. Ishikawa: Rule extraction by successive regularization. Neural Networks *13* (2000), 1171–1183.
[31] H. Lu, R. Setiono, and H. Liu: Effective data mining using neural networks. IEEE Trans. Knowledge and Data Engrg. *8* (1996), 957–961.

[32] F. Maire: Rule-extraction by backpropagation of polyhedra. Neural Networks *12* (1999), 717–725.
[33] R. McNaughton: A theorem about infinite-valued sentential logic. J. Symbolic Logic *16* (1951), 1–13.
[34] S. Mitra, R. K. De, and S. K. Pal: Knowledge-based fuzzy MLP for classification and rule generation. IEEE Trans. Neural Networks *8* (1997), 1338–1350.
[35] S. Mitra and Y. Hayashi: Neuro-fuzzy rule generation: Survey in soft computing framework. IEEE Trans. Neural Networks *11* (2000), 748–768.
[36] S. Muggleton: Inductive Logic Programming. Academic Press, London 1992.
[37] D. Mundici: A constructive proof of McNaughton's theorem in infinite-valued logic. J. Symbolic Logic *59* (1994), 596–602.
[38] H. Narazaki, T. Watanabe, and M. Yamamoto: Reorganizing knowledge in neural networks: An exploratory mechanism for neural networks in data classification problems. IEEE Trans. Systems Man Cybernet. *26* (1996), 107–117.
[39] D. Nauck, U. Nauck, and R. Kruse: Generating classification rules with the neuro-fuzzy system NEFCLASS. In: Proc. Biennial Conference of the North American Fuzzy Information Processing Society NAFIPS'96, 1996, pp. 466–470.
[40] V. Novák and I. Perfilieva: Some consequences of herbrand and McNaughton theorems in fuzzy logic. In: Discovering World with Fuzzy Logic: Perspectives and Approaches to Formalization of Human-Consistent Logical Systems (V. Novák and I. Perfilieva, eds.), Springer–Verlag, Heidelberg 1999, pp. 271–295.
[41] V. Novák, I. Perfilieva, and J. Močkoř: Mathematical Principles of Fuzzy Logic. Kluwer Academic Publishers, Dordrecht 1999.
[42] I. Perfilieva: Neural nets and normal forms from fuzzy logic point of view. Technical Report, Institute for Research and Applications of Fuzzy Modelling, 2001.
[43] I. Perfilieva: Normal forms for fuzzy logic functions and their approximation ability. Fuzzy Sets and Systems *124* (2001), 371–384.
[44] I. Perfilieva: Logical approximaion. Soft Computing *7* (2003), 73–78.
[45] I. Perfilieva: Normal forms in BL-algerbra and their contribution fo universal approximation of functions. Fuzzy Sets and Systems *143* (2004), 111–127.
[46] I. Perfilieva: Normal forms in BL and ŁΠ algebras of functions. Soft Computing *8* (2004), 291–298.
[47] I. Perfilieva and V. Kreinovich: A new universal approximation result for fuzzy systems, which reflects CNF-DNF duality. Internat. J. Intelligent Systems *17* (2002), 1121–1130.
[48] L. Polkowski: Rough Sets. Mathematical Foundations. Physica–Verlag, Heidelberg 2002.
[49] J. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco 1992.
[50] R. Setiono: Extracting rules from neural networks by pruning and hidden unit splitting. Neural Computation *9* (1997), 05–225.
[51] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich: The truth will come to light: Directions and challenges in extracting rules from trained artificial neural networks. IEEE Trans. Neural Networks *9* (1998), 1057–1068.
[52] G. G. Towell and J. W. Shavlik: Extracting refined rules from knowledge-based neural networks. Mach. Learning *13* (1993), 71–101.
[53] E. Triantaphyllou and G. Felici (eds.): Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques. Kluwer Academic Publishers, Dordrecht 2003.
[54] H. Tsukimoto: Extracting rules from trained neural networks. IEEE Trans. Neural Networks *11* (2000), 333–389.

[55] M. L. Wong and K. S. Leung: Data Mining Using Grammar Based Genetic Programming and Applications. Kluwer Academic Publishers, Dordrecht 2000.
[56] C. Zhang, S. Zhang, and B. E. Heymer: Association Rule Mining: Models and Algoritms. Springer–Verlag, Berlin 2002.

Martin Holeňa, Institute of Computer Science – Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 182 07 Praha 8. Czech Republic.
e-mail: martin@cs.cas.cz