Computational Intelligence Methods for Rule-Based Data Understanding

WŁODZISŁAW DUCH, RUDY SETIONO, SENIOR MEMBER, IEEE, AND JACEK M. ŻURADA, FELLOW, IEEE

Contributed Paper

In many applications, black-box prediction is not satisfactory, and understanding the data is of critical importance. Typically, approaches useful for understanding of data involve logical rules, evaluate similarity to prototypes, or are based on visualization or graphical methods. This paper is focused on the extraction and use of logical rules for data understanding. All aspects of rule generation, optimization, and application are described, including the problem of finding good symbolic descriptors for continuous data, tradeoffs between accuracy and simplicity at the rule-extraction stage, and tradeoffs between rejection and error level at the rule optimization stage. Stability of rule-based description, calculation of probabilities from rules, and other related issues are also discussed. Major approaches to extraction of logical rules based on neural networks, decision trees, machine learning, and statistical methods are introduced. Optimization and application issues for sets of logical rules are described. Applications of such methods to benchmark and real-life problems are reported and illustrated with simple logical rules for many datasets. Challenges and new directions for research are outlined.

Keywords*—*Data mining, decision support, decision trees, feature selection, fuzzy systems, inductive learning, logical rule extraction, machine learning (ML), neural networks, neurofuzzy systems.

I. INTRODUCTION

Prediction of the stock market, or of the weather changes in the next few days, is a goal in itself. Black-box statistical approaches to data analysis that offer the best fit to the data are a satisfactory solution for such problems. Prediction of climate changes is also very important, but understanding of the factors that facilitate the changes is of even greater importance. These changes could be summarized by a rule: IF fossil fuels are burned, THEN the climate warms up. Formulation of understandable rules derived from analysis of data is not the same as creating predictive models of data.

Many methods of data analysis devised in pattern recognition, neural networks, evolutionary computation, and related fields are aimed mainly at building predictive data models, adapting internal parameters of the data models to account for the known (training) data samples and allowing for predictions to be made on the unknown (test) data samples. For example, naive Bayesian methods in statistics typically fit Gaussian distributions to data. Linear discrimination methods search for hyperplanes that separate different classes. Support vector machines (SVMs) provide nonlinear hypersurfaces for the same purpose, while multilayered perceptron (MLP) neural networks combine many sigmoidal basis functions adjusting internal parameters to create, using training data, vector mappings from the input to the output space. Discovery of class structures, interesting association patterns, sequences, or causal relationships has never been an explicit goal in designing such methods [1], [2].

Predictive nonparametric classification and approximation methods frequently achieve high accuracy using a large number of numerical parameters in a way that is incomprehensible to humans. This leads to several dangers. When the number of parameters is of the order of the number of data vectors, predictive models may easily overfit the data. In some cases, even an abundance of data will not prevent overfitting [3]. Many irrelevant attributes may contribute to the final solution. Combining predictive models with *a priori* knowledge about the problem is usually difficult. Therefore, the use of the black-box models in expert systems that require systematic reasoning and offer explanations

Manuscript received February 16, 2003; revised February 4, 2004. The work of W. Duch was supported in part by the Polish Committee for Scientific Research under Grant 8 T11C 006 19. The work of J. M. Żurada was supported in part by the Systems Research Institute, Polish Academy of Science.

W. Duch is with the Department of Informatics, Nicolaus Copernicus University, Toruń 87-100, Poland, and was also with School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: duch@ieee.org).

R. Setiono is with the School of Computing, National University of Singapore, Singapore 119260 (e-mail: rudys@comp.nus.edu.sg).

J. M. Żurada is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY 40292 USA (e-mail: j.zurada@ieee.org).

Digital Object Identifier 10.1109/JPROC.2004.826605

0018-9219/04$20.00 © 2004 IEEE

of their recommendations may not be possible. In novel situations, predictions of the black-box models may be quite unreasonable, since there is no way to control and test the model in the areas of the future space that are far from the training data. In safety-critical domains, such as medical, industrial, or financial applications, such risks may not be acceptable.

The *a priori* knowledge about a problem to be solved is frequently given in a symbolic, rule-based form. Extraction of knowledge from data, combining it with available symbolic knowledge, and refining the resulting knowledge-based expert systems is a great challenge for computational intelligence. Reasoning with logical rules is more acceptable to human users than the recommendations given by black box systems [4], because such reasoning is comprehensible, provides explanations, and may be validated by human inspection. It also increases confidence in the system, and may help to discover important relationships and combination of features, if the expressive power of rules is sufficient for that.

Machine learning (ML) started as a subfield of artificial intelligence (AI), setting as its explicit goal the formulation of symbolic inductive methods (i.e., methods that learn from examples) [5]. These methods were supposed to discover rules that could be expressed in natural language, and would be similar to those a human expert might create. Since achieving this type of understanding is not always feasible, ML has broadened in recent years to include all methods that learn from data.

Verification of understanding text that is read, or multimedia data that is viewed or listened to, may be done by questioning, summarizing, and other verbal techniques. It does not mean that the "inner models" in the brain are based on logical constructions that are used for effective communication. Symbolic description is not the only way to understand data. In fact, cognitive psychology experiments show that human categorization is based on memorization of examples and creation of prototypes that are abstractions of these examples, rather than on logical rules defining natural objects in some feature spaces [6]. "Intuitive understanding" is based on experience, i.e., on memorized examples of patterns combined with various similarity measures that allow for their comparison and evaluation. Decision borders between different categories produced in this way may be quite complex and difficult to describe using linguistic statements.

Visualization provides another way of understanding data—a single picture may be worth a thousand words. Visualization of various signals carrying multidimensional information is frequently used in military, medical, and industrial applications. Visualization forms a basis of the exploratory data analysis (EDA) that tries to uncover underlying data structure, detect outliers and anomalies, and find important variables [7], [8]. Experts are able to understand the data simply by inspecting such visual representations. Visualization of neural networks outputs and activities of hidden layer allows to understand better mappings they perform [9]. A special form of visualization is afforded by graphical methods that are aimed at the representation of the relationships between different elements of the problem description [10]. Bayesian belief networks are a good example of such graphical models.

The best explanation of the data obviously depends on the type of problem and the intention of the user, as well as the type of questions and explanations that are commonly accepted in a given field. Among many methods of data understanding, this paper focuses on classification rules in their simplest, propositional form, derived from datasets that contain structured information. We shall assume that a set of symbolic or continuous-valued predicate functions has been defined for some objects, thus providing values of attributes (features) for categorization of these objects. The intention here is to understand the class structure of these objects. Even this limited focus area has so many aspects that it is difficult to review it in one paper. There are hundreds of ways one can combine neural, fuzzy, similarity-based, rough, inductive, clusterization, optimization, genetic, and other evolutionary techniques to find and optimize sets of logical rules. Because there is an overabundance of algorithms, only those computational intelligence techniques that proved to be directly useful to data understanding have been presented here. A long section presenting various applications not only illustrates the usefulness of some methods, but also provides a gauge to evaluate progress in this field.

In the next two sections, types of propositional logical rules are discussed, followed by the discussion of linguistic variables and the expressive power of rules. Section IV describes decision trees for rule generation, Section V presents ML approaches, and Section VI discusses neural network methods for logical rule extraction. This is followed by optimization of the sets of rules, various tradeoffs involved, and application of logical rules for calculation of probabilities in Section VII. An illustrative example to the benchmark Iris data is given in Section VIII, and applications of various rule-extraction methods to the analysis of benchmark and real-world datasets are described in Section IX. Challenges and new directions of research are discussed at the end of the paper.[1]

II. Propositional Logic Rules

This section discusses various types of logical rules in the context of decision borders they create in multidimensional feature spaces. Standard crisp propositional IF ... THEN rules are the simplest and most comprehensible way of expressing knowledge; therefore, they are discussed first.

Although the form of propositional rules may differ, they always partition the whole feature space $\mathcal{X}$ into some subspaces. A general form of a crisp rule is

$$\text{IF } \mathbf{X} \in \mathcal{X}^{(i)} \text{ THEN Class}(\mathbf{X}) = C_k. \tag{1}$$

[1]The authors have included excerpts from their earlier paper [11] and [151], as they felt that this inclusion would be beneficial to the readers who are not necessarily specialists in computational intelligence techniques, and that doing so would enhance the tutorial value and completeness of this survey paper. Additional need for the inclusion has arisen because of the nonoverlapping readership of the Proceedings of the IEEE and the specialized literature sources cited in this work.

If $\mathbf{X}$ belongs to the subspace $\mathcal{X}^{(i)}$, then it should be assigned a class label C_k. A fuzzy version of this rule is a mapping from the $\mathcal{X}$ space to the space of fuzzy class labels. However, without certain restrictions on the shapes of decision borders, such rules can be difficult to comprehend.

A fruitful way of looking at sets of logical rules is to treat them as a classification model that provides an approximation to the posterior probability $p(C_k|\mathbf{X};M)$. Given the sample vector $\mathbf{X}$ and a model M based on the set of rules, the goal is to provide an estimate of probability that $\mathbf{X}$ belongs to class C_k. Crisp logic rules should give precise "yes" or "no" answers; therefore, $p(C_k|\mathbf{X};M) \in \{0,1\}$.

The condition part of a rule is defined by a conjunction of logical predicate functions $L_j(\mathbf{X}) \in \{F,T\}$. In the most common case, predicate functions are tests on a single attribute $L_{ij}(X_i) = T$ if feature X_i has values that belong to a subset (for discrete features) or to an interval $X_i \in [X_{ij-}, X_{ij+}]$. The index j enumerates here intervals or (fuzzy) subsets for attribute X_i associated with tests L_{ij}. The conjunction of such conditions for several attributes defines a hyperrectangular area covering a part of the feature space where the rule is assumed to be true, expressed as

$$L_{i_1 j_1}(X_{i_1}) \wedge L_{i_2 j_2}(X_{i_2}) \ldots \text{THEN Class}(\mathbf{X}) = C_k. \tag{2}$$

Some algorithms [4], [11] generate rules for which hyperrectangular covering areas may overlap. Classification probabilities in the overlapping region $\mathcal{R}$ may be estimated by taking $p(C_k|\mathbf{X};M) = N(C_k, \mathbf{X} \in \mathcal{R})/N(\mathbf{X} \in \mathcal{R})$, that is, the number of training vectors from the class C_k in the region $\mathcal{R}$ divided by the number of all training vectors falling into this region. Other ways of resolving conflicts are based on voting procedures or selecting rules with the lowest false positive rate [12]. Decision trees and algorithms that use discretized data explicitly avoid such overlaps. Most decision trees are based on simple tests $L_{ij}(X_i) < T_{ij}$. For ordered variables, each test defines a hyperplane perpendicular to X_i and intersecting its axis at T_{ij}. Since the tests are done in a hierarchical manner, the feature space is partitioned recursively into hyperrectangles within hyperrectangles (see Fig. 1). All rules obtained from decision trees are, therefore, nonoverlapping, sharing antecedents in a hierarchical manner.

Fuzzy set theory is based on real-valued predicate functions $L_{ij}(X_i)$ called membership functions. The usual interpretation is that X_i belongs to a fuzzy set S_{ij} to a degree $L_{ij}(X_i)$. Instead of input values X_i, membership values $L_{ij}(X_i)$ in several fuzzy sets are used in rule conditions. In many cases, these fuzzy sets have reasonable interpretation. For example, if the membership functions are $L_1(X) = \text{low pressure}(X) = 0.25$, $L_2(X) = \text{normal pressure}(X) = 0.65$ and $L_3(X) = \text{high pressure}(X) = 0.05$, one may say that the pressure is a bit low. Using logical predicates for the variable "pressure" leads to a sudden jump, from one interval to the adjacent one, when a small change in the input value is made. Fuzzy logic removes such discontinuities. Predicates based on intervals $L_{ij} = T$ iff $X_{ij-} \leq X_i \leq X_{ij+}$ lead to

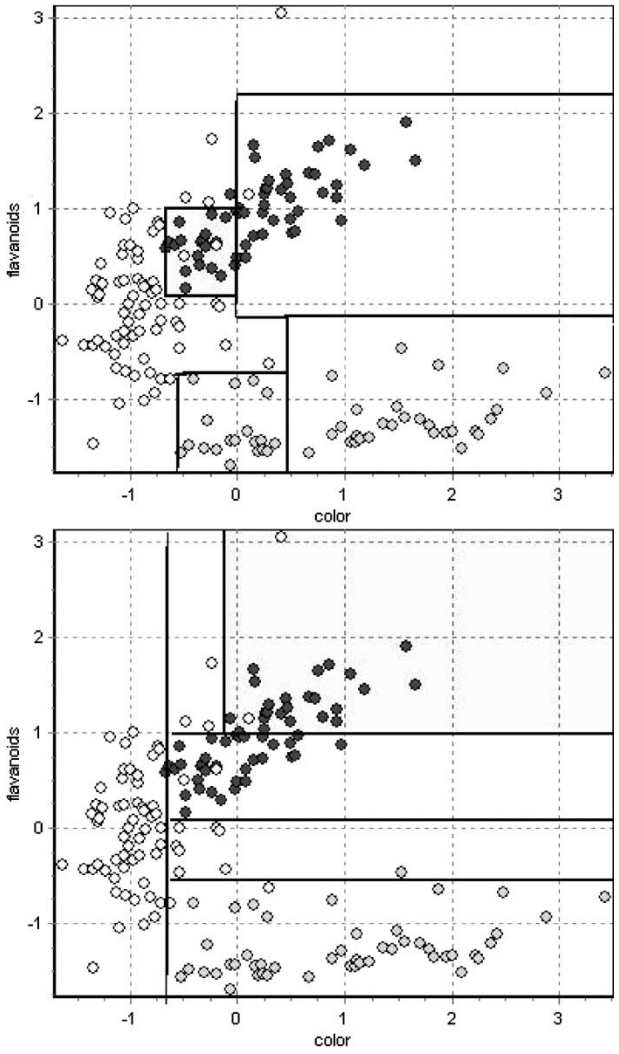

Fig. 1. Three kinds of wine, with over 50 samples for each kind displayed using *color* and *flavanoid content* features. Decision borders for propositional crisp logical rules separating wine types divide the feature space into hyperrectangles, with the ELSE condition covering the remaining space (upper figure). Decision trees partition the space in a hierarchical manner (lower figure).

hyperrectangular decision borders, while soft localized or semilocalized membership functions provide more flexible decision borders.

Flexibility of the fuzzy approach depends on the choice of membership functions. Fuzzy logic classifiers frequently use a few membership functions per input feature [13]–[15]. Triangular membership functions provide oval decision borders, similar to those provided by Gaussian functions (see Fig. 2). In fact, each fuzzy rule may be represented as a node of a network that processes input vectors. The resulting basis function network is formally equivalent to an inference system based on fuzzy rules [14]. Triangular membership functions may be regarded as a piecewise linear approximation to Gaussian membership functions, while trapezoidal membership functions are similar approximations to the soft trapezoid functions obtained from combinations of two sigmoidal transfer functions (see next section).

The fuzzy set theory [13]–[16] gives only a formal definition of membership function $L_{ij}(X_i)$ and $X_i \in S_{ij}$ relation,

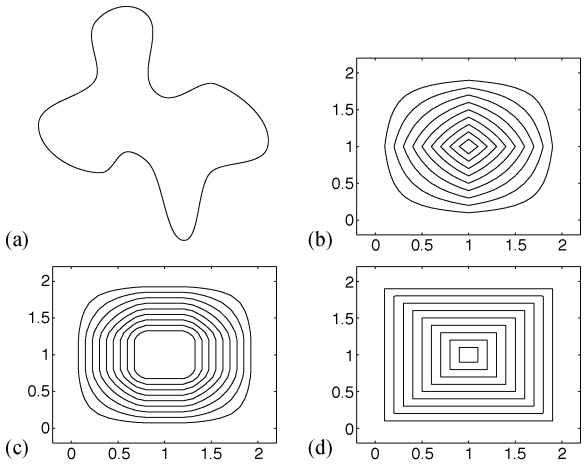

Fig. 2. Shapes of decision borders for: (a) general clusters; (b) fuzzy rules (using product of membership function); (c) rough rules (trapezoidal approximation); and (d) crisp logical rules.

but the precise meaning of fuzzy rules and the $\in$ operator is not determined. One natural interpretation is based on the similarity of $\mathbf{X}$ to the prototypes typical for the C_k class, but at least four other interpretations may be distinguished [17]. Membership functions should reflect some properties of the data distribution, such as position of clusters in the data or similarity to known prototypes.

The conjunctive form of the rule conditions, as in (2), is most common, but other ways to combine conditions may also be used. In some applications, it is sufficient that at least M out of N possible conditions of a rule are true. These M-of-N rules are easy to define if the predicate functions return integer values zero or one, instead of logical values true or false. The antecedent of the rule (2) becomes

$$\text{IF} \sum_{i=1}^{N} L_{\text{ij}}(X_i) \geq M \text{ THEN Class}(\mathbf{X}) = C_k \tag{3}$$

where for each feature i, an interval (linguistic term) $j = j(i)$ is picked up.

Such rules are implemented in a natural way by a threshold function $\Theta(\mathbf{W} \cdot \mathbf{X} - M)$, where all X_i are binary, $W_i = 1$, and $\Theta(x) = 0$ for $x < 0$, or $\Theta(x) = 1$ for $x \geq 0$. In neural network terminology, such threshold function is called a logic neuron (see Section VI), and if W_i, X_i are arbitrary real numbers and the threshold function is replaced by a smooth function of sigmoidal shape (for example, hyperbolic tangent), it is called a perceptron. Threshold functions create rules that are equivalent to an alternative of many conjunctive terms. For example, if $M = N/2$ (at least half of the conditions should be true), then the number of equivalent conjunctive terms is 2^{N-1}. The decision borders defining the classes in the feature space are still hyperplanes perpendicular to the axes, but the shapes of the regions where the M-of-N rule is true may be quite complex and difficult to analyze. If the predicate functions $L_{\text{ij}}(X_i) \in [0, 1]$ return continuous values corresponding to the degree of fulfillment of some tests on the attribute X_i, the weighted M-of-N rule becomes equivalent to the decision of a perceptron, with a hyperplane defining the decision border.

M-of-N rule conditions are based on classical conjunctions, although the number of terms may be quite large. Fuzzy logic uses general functions $T(A, B)$, called T-norms, to replace $A \wedge B$ when two conditions $A = L_i(X_i)$, $B = L_j(X_j)$ are combined; $L_i(X_i)$, $L_j(X_j)$ are here real-valued fuzzy membership functions, such as Gaussian or triangular-shape functions. If $A, B \in [0, 1]$ are continuous, then the product $A \cdot B$, $\min(A, B)$, or $\max(0, A + B - 1)$ may be taken as the T-norm. We shall not go in this direction further, since the interpretability of rules using more sophisticated T-norms is rather difficult to grasp in an intuitive way.

Another way to generalize propositional rules is to admit predicate functions $L_j(\mathbf{X})$ that perform tests on more than one attribute of $\mathbf{X}$. For example, a distance $\|\mathbf{X} - \mathbf{R}\| < \theta(\mathbf{R})$, between vector $\mathbf{X}$ and a prototype $\mathbf{R}$, lower than the threshold $\theta(\mathbf{R})$, may be used to determine if $L_j(\mathbf{X}) = T$. Such prototype-based rules are quite natural in some applications and may still be intuitively easy to understand. Again, instead of logical predicates, some degree of fulfillment of such condition may be introduced. For example, the logistic function $\sigma(x) = 1/(1+e^{-x})$ defines a fuzzy distance-based rule

$$L_j(\mathbf{X}) = \sigma\left(\|\mathbf{X} - \mathbf{R}\| - \theta(\mathbf{R})\right) \tag{4}$$

producing a fuzzy degree of truth value $0 < L_j(\mathbf{X}) < 1$.

The shapes of decision borders provided by such conditions depend on the distance functions used. For continuous features and Euclidean distance functions, hyperellipsoidal decision borders are obtained (for normalized features they are spherical). Using L_∞ norm (Chebyschev norm) $\|(\mathbf{X} - \mathbf{R})\|_\infty = \max_i |X_i - \mathbf{R}_i| < \theta(\mathbf{R})$ leads to hyperrectangular decision borders [18] with prototype $\mathbf{R}$ in the center.

Although fuzzy rules may be written in the same form as (2), the logical $\wedge$ operator is now replaced by the appropriate T-norm, usually a product of membership functions for individual features. A rule $\mathcal{R}$ is fulfilled to the degree $L_{\mathcal{R}}(\mathbf{X})$

$$L_{\mathcal{R}}(\mathbf{X}) = \prod_{(ij)\in\mathcal{R}} L_{\text{ij}}(X_i) \tag{5}$$

where the product includes all X_i attributes that appear in the rule condition, and all membership functions L_{ij} of fuzzy subsets of X_i attribute used in the definition of the rule $\mathcal{R}$. Summing the degrees of fulfillment of all rules $L_{\mathcal{R}(k)}(\mathbf{X})$ associated with class C_k and dividing over the sum of degrees of fulfillment of all rules gives an estimation of classification probability

$$p(C_k|\mathbf{X}; M) = \frac{\sum_{\mathcal{R}(k)} L_{\mathcal{R}(k)}(\mathbf{X})}{\sum_{\mathcal{R}} L_{\mathcal{R}}(\mathbf{X})}. \tag{6}$$

Some Gaussian classifiers [realized, for example, by radial basis function (RBF) networks [14]], equivalent to fuzzy systems with Gaussian membership functions, sum the network outputs assigned to class C_k, dividing the result by the sum of all outputs to normalize it. These estimates are not "true probabilities," but they give an idea of how strongly the rules support the assignment of vector $\mathbf{X}$ to different classes C_k.

The conclusion of a rule may also be generalized to cover fuzzy labels or real-valued labels $C_k \in [0,1]$. For example, rules predicting color may use the continuous value corresponding to the light wavelength, or cluster the results around some prototypes, such as "green" or "red."

With continuous functions determining conditions and continuous labels as conclusions, rules become general mappings of object features $\mathbf{X}$ to some labels, $M(\mathbf{X}) = C$. For example, spectral properties of light sources and surfaces reflecting them are mapped into colors. The feature space $\mathcal{X}$ is partitioned in a fuzzy or crisp way by a set of membership functions $L^{(j)}(\mathbf{X}) \in [0,1]$ defined for all $\mathbf{X} \in \mathcal{X}$, for example, a set of overlapping Gaussians for the light spectrum. A fuzzy subspace $\mathcal{X}^{(i)}$ contains all vectors with nonzero degree of membership $L^{(i)}(\mathbf{X}) > 0$. Since multivariate mappings are difficult to interpret, understanding the data using rules may be regarded as an attempt to discretize the mapping in some way.

Rough set theory [19], [20] can also be 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 C_k with probability $p(C_k|\mathbf{X};M) = 1$, while the upper approximation covers all instances which have a nonzero chance $(p(C_k|\mathbf{X};M) > 0)$ to belong to this class. 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, i.e., the same shapes of decision borders as those obtained by fuzzy systems with such membership functions. The crisp form of logical rules is obtained when trapezoidal membership functions are changed into rectangular functions. These rectangles allow for the definition of logical linguistic variables for each feature by intervals or sets of nominal values (see Fig. 2).

Decision borders provided by the crisp and fuzzy rules frequently do not allow for a good approximation with a small number of rules. From the perspective of accuracy and simplicity, the ability to deal with oblique distributions of data may be more important than softer decision borders. Using combinations of input features makes the meaning of rules based on such new features difficult to comprehend ("mixing apples with oranges"). Another form of incomprehensible rules is obtained from a union of half-spaces defined by hyperplanes, forming a convex, polyhedral shape. "... [T]o what extent are fuzzy classifiers useful as *fuzzy*, and at which point do they turn into black boxes? ... Practice has shown so far that trying to reach the accuracy of good nonfuzzy model by a fuzzy one is likely to require more time and resources than for building up the initial nonfuzzy classifier" [21]. The design of a rule-based system is, thus, always a compromise between the flexibility of decision borders and the comprehensibility of the rules.

Although individual fuzzy, rough, and neurofuzzy systems differ in their approach to logical rule discovery, their ultimate capability depends on the decision borders that they provide for classification. A natural category may have quite complex shape in feature space. Geometrical description and visualization of clusters may be used to query relational databases [22]; here we are interested in the automatic discovery of such descriptions. From a methodological point of view, one should always first try the simplest models based on crisp logic rules, and only if they fail, more complex forms of rules should be attempted. Neural networks with nodes that implement separable transfer functions (i.e., calculate products of functions, one for each feature) in a single hidden layer, are capable of creating the same decision borders as crisp, fuzzy, or rough set rule sets [23].

Propositional logic has in itself a limited expressive power and may be used only in domains where attribute-value language (i.e., vector feature space description) is sufficient to express knowledge. Complex objects, represented by graphs or multisets, should be treated with first-order or higher order logic [24], [25]. Since even the full first-order logic is computationally difficult to implement, various restrictions have been proposed to make the process of rule discovery computationally effective [26].

III. Linguistic Variables

Logical rules (as do other attempts to verbalize knowledge) require symbolic inputs, called linguistic variables. This implies that the input data have to be quantized, i.e., features defining the problem should be identified, and their subranges (sets of symbolic values, integer values, or continuous intervals) labeled. For example, in crisp logic, a variable "size" has the value "small" if the continuous variable X_i measuring size falls in some specified range $X_i \in [a,b]$. Using one continuous input variable, several binary (logical) variables may be created: size small $= \delta$(size,small) equal to one (true) only if variable size has the value small.

A. Types of Linguistic Variables

Two types of linguistic variables are in use. Universal, *context-independent* variables are created by partitioning the range of feature values and using the resulting linguistic variables without any modification in logical rules. They are identical in all regions of the feature space. Using such variables makes it easy to avoid overlapping of the decision regions in different rules. Defining, for example, three triangular membership functions per attribute—$L_1(X_i)$, $L_2(X_i)$, $L_3(X_i)$—rules for combinations

$$\text{IF}\,(L_{i_1}(X_1) \wedge L_{i_2}(X_2) \ldots \wedge L_{i_N}(X_N))\ \text{THEN Class} = C_k$$

are sought [15], with $i_j = 1, 2, 3$. Unfortunately, the number of combinations grows exponentially with the number of attributes (here as 3^N), restricting the applications of context-independent partitioning to a small number of dimensions. Although attempts to overcome the combinatorial explosion by changing the form of fuzzy rules have been reported [27], [28], covering a complex data cluster may require a large number of such membership functions.

Context-dependent linguistic variables may be different in each rule [29]. Their definition takes into account interaction between linguistic variable in the process of rule forma-

tion, optimizing intervals in each rule. Low tire pressure for a bicycle is different than for a car or a truck. For example, taking $L_{11}(X_1) =$ broad iff $1 \leq X_1 \leq 4$, $L_{12}(X_1) =$ narrow iff $2 \leq X_1 \leq 3$, and $L_{21}(X_2) =$ small iff $1 \leq X_2 \leq 2$, $L_{22}(X_2) =$ large iff $3 \leq X_2 \leq 4$, two simple rules

$$\begin{aligned}&\text{IF } (L_{11}(X_1) = \text{broad} \wedge L_{21}(X_2) = \text{small})\\ &\quad \text{THEN } C = \text{great}\\ &\text{IF } (L_{12}(X_1) = \text{narrow} \wedge L_{22}(X_2) = \text{large})\\ &\quad \text{THEN } C = \text{great}\\ &\text{ELSE } C = \text{so-so}\end{aligned}$$

would be more complex if written using linguistic variables that partition X_1 into distinct or just partially overlapping subsets. In the context of linguistic variable $X_2 =$ large, linguistic value $X_1 =$ narrow, a subset of broad should be used. Instead of using a fixed number of linguistic variables per feature, one should use context-dependent linguistic variables, optimized for each rule.

Depending on the type of variable X, the predicate function $L(X)$ defining a linguistic variable may have a different interpretation. If X is the wavelength of light, $X \in [600\text{ nm}, 700\text{ nm}]$, then Color$(X)$ is red, i.e., logical condition Color$(X) =$ Red is true. One may also introduce predicates for each color defined by logical functions Color green(X), Color red(X), Color blue(X), etc. Such logical predicate functions map symbolic or real values of X into binary 0, 1 or logical *false, true* values.

B. Methods for Creating Linguistic Variables

Determination of intervals defining linguistic variables for real-valued attributes gives rise to a good discretization of continuous data. The simplest way to select initial linguistic variables is to analyze histograms obtained by displaying data for all classes for each feature. Histograms should be smoothed, for example, by assuming that each feature value is really a Gaussian or a triangular fuzzy number (kernel smoothing techniques are discussed in [30]). Unfortunately, histograms for different classes frequently overlap strongly, limiting the applicability of this method for finding linguistic variables (as shown in Fig. 7 later).

Methods that are useful for creating linguistic variables draw inspiration from different fields [31]. Global discretization methods are independent of any rule-extraction method, treating each attribute separately. Local discretization methods are usually embedded in rule-extraction algorithms, performing discretization using a subset of all data only (the subset covered by a decision tree node or a given network node). Global methods may be based on: 1) searching intervals that contain vectors from a single class [32]; 2) entropy or information theory to determine intervals that have low entropy or high mutual information with class labels [33]; 3) univariate decision trees to find the best splits for a single feature [34]; 4) latent variable models [35]; or 5) Chi-square statistics to merge intervals [36].

A discretization algorithm based on a separability criterion [37], described below, creates a small number of intervals (or subsets) with high information content. The best "split value" for an open interval should separate the maximum number of pairs of vectors from different classes. Among all split values that satisfy this condition, the one that separates the smallest number of pairs of vectors belonging to the same class is selected. The criterion is applicable to both continuous and discrete features. Since one feature is treated at a time, the minimization process is fast. The split point for continuous features is a real number s, while for discrete features it is a subset of the set of alternative values of the feature. In all cases, the *left side (LS)* and the *right side (RS)* of a split value s is defined by a test $f(\mathbf{X}, s)$ for a given dataset $\mathcal{D}$

$$\begin{aligned}\text{LS}(s, f, \mathcal{D}) &= \{\mathbf{X} \in \mathcal{D} : f(\mathbf{X}, s) = T\}\}\\ \text{RS}(s, f, \mathcal{D}) &= \mathcal{D} - \text{LS}(s, f, \mathcal{D})\end{aligned} \tag{7}$$

where a typical test $f(\mathbf{X}, s)$ is true if the selected feature $X_i < s$ or (for discrete feature) $X_i \in \{s\}$. The *separability of a split value* (SSV), or *of a subset* s, is defined for a given test f as

$$\begin{aligned}\text{SSV}(s,f) = 2\sum_{k=1}^{K} &|\text{LS}(s, f, \mathcal{D}) \cap \mathcal{D}_k| \cdot |\text{RS}(s,f,\mathcal{D}) \cap (\mathcal{D} - \mathcal{D}_k)|\\ &- \sum_{k=1}^{K} \min\left(|\text{LS}(s, f, \mathcal{D}) \cap \mathcal{D}_k|, |\text{RS}(s, f, \mathcal{D}) \cap \mathcal{D}_k|\right)\end{aligned} \tag{8}$$

where K is the number of classes, $\mathcal{D}_k$ is the set of data vectors $\mathbf{X} \in \mathcal{D}$ that belong to the class C_k, and $|\mathcal{F}|$ is the number of elements in the set $\mathcal{F}$. Points outside of the feature value range have the SSV equal to zero, while separability for all other values is positive. For two separable classes with $N(C_1)$, $N(C_2)$ elements, SSV has a maximum value $2N(C_1)N(C_2) - \min(N(C_1), N(C_2))$. For each feature with at least two different values, at least one split value of maximal separability exists. If maximum separability is reached for several split values close to each other, the split value closest to the average is selected.

The separability criterion can be used to discretize a continuous feature to find both context-independent and context-dependent linguistic variables. In the first case, it is used for each feature independently, until the increase in separability due to new splits is sufficiently small, or when a sufficient number of linguistic variables have been obtained. If all split values for a given feature have a very low separability, the feature either does not contribute any information or should be taken into account in conjunction with the discretization of another feature. This leads to the context-dependent linguistic variables that are obtained from a partial decision tree built using the separability criterion.

A search for the best separability of a pair or a combination of several features is performed quite efficiently by using hierarchical partitioning with best-first search for the best split points, or using more expensive beam search techniques that escape local minima. For a pair of the features, the beam search complexity is quadratic in the number of split values considered, enabling in practice exhaustive search. Searching for all feature split values at the same time takes into account mutual interactions of features. Therefore, it may significantly improve results. However, since the search complexity is high, a small width of the beam search should be selected to make it practical.

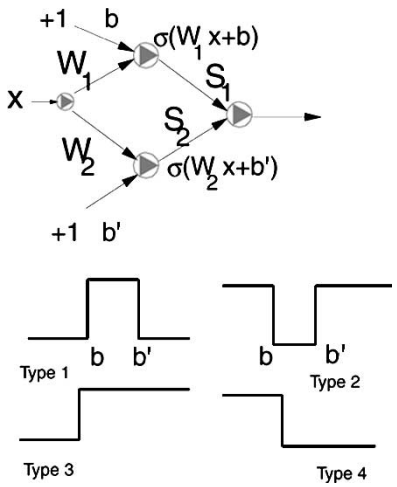

Fig. 3. Construction of a linguistic unit converting continuous inputs to linguistic variables. This unit calculates a function $F(X) = S_1\sigma(W_1X + b) + S_2\sigma(W_2X + b')$. Type 1 functions, equivalent to $\Theta(X - b) - \Theta(X - b')$, are obtained for $W_1 = W_2 = S_1 = +1, S_2 = -1$, and implement logical condition $L(X) =$ True iff $X \in [b, b')$. Type 2 functions are negation of Type 1 functions, and are obtained, for example, with $W_1 = -1, W_2 = S_1 = S_2 = +1$. The remaining two types of functions are obtained for $S_2 = 0$ or for b' outside of the data range.

C. Feature-Space Mapping Networks

Local discretization methods may also use neurofuzzy algorithms, adjusting adaptive parameters of a network to model probability density functions. RBF networks with Gaussian membership functions may be used for extracting fuzzy rules. A feature space mapping (FSM) is a constructive neural network [23], [38], [39] that estimates the posterior probability $p(C|\mathbf{X}; M)$. Nodes of this network calculate localized, separable transfer functions $F(\mathbf{X}) = \prod_i F_i(X_i)$ providing linguistic variables $F_i(X_i)$. Crisp decision regions are obtained by using rectangular transfer functions $F_i(X_i) = \Theta(X_i - X_{i-}) - \Theta(X_i - X_{i+})$ where $\Theta(X)$ is the step function equal to one for $X \geq 0$ (see Fig. 3). If this is not sufficient, Gaussian, trapezoidal, and bicentral combinations of sigmoidal functions [40] or other separable transfer functions may be used. A bicentral type of function is constructed as the difference of two sigmoidal functions $F_i(X_i) = \sigma(X_i - X_{i-}) - \sigma(X_i - X_{i+})$ or as the product of pairs of sigmoidal functions $\sigma(X_i - X_{i-})(1 - \sigma(X_i - X_{i+}))$ for all features. For logistic sigmoidal function $\sigma(X) = 1/(1 + e^{-X})$, the two types of bicentral functions are identical after normalization [11]. These functions have soft trapezoidal shapes, providing natural generalization of rectangular crisp logic membership functions.

The FSM network is initialized using a decision tree, or a clustering method based on dendrograms [38]. A small number of network functions $F^{(k)}(\mathbf{X})$ that have nonzero values for the training data are set up. Each network node covers a cluster of input vectors. The initialization process is robust and may lead to reasonable intervals for the initial linguistic variables without any training. The training procedure changes the positions and dispersions of the functions to cover the data more precisely. The node that on average has the largest output for all training vectors covers the largest number of input vectors. This node, assigned to a majority class, corresponds to the most general logical rule. Training of the network, or adaptation of $F^{(k)}(\mathbf{X})$ node parameters, is equivalent to learning context-dependent membership functions (factors $F_i^{(k)}(X_i)$) and fuzzy conjunctive rules (products of these factors).

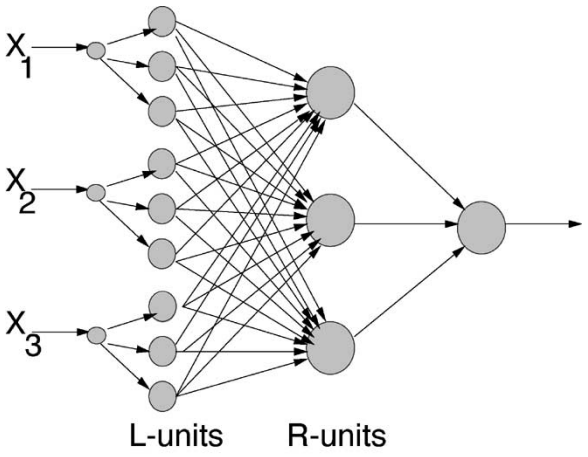

Fig. 4. MLP network with linguistic (L) and rule (R) units. An additional aggregation layer may be added between the input and L-units.

If bicentral functions are used and the slope of the sigmoidal function $\sigma(X)$ is slowly increased during training, the soft trapezoidal shapes of membership functions for each feature are transformed in a smooth way into rectangles, providing crisp linguistic variables. In this case, for each network node, the component $F_i^{(k)}(X_i)$ is defined by an interval $[b_k, b'_k]$ corresponding to a linguistic variable. This interval is adjusted to cover the largest number of vectors that belong to the class associated with the $F^{(k)}(\mathbf{X})$ node. Linguistic variables that are always true (factors $F_i^{(k)}(X_i) \approx 1$ for all X_i within the data range) may be dropped, since they do not carry any information. Nodes that cover only a few training vectors are removed and nodes that cover many training vectors are optimized. If necessary, more nodes are added to the network, playing the role of specialized logical rules [39].

Linguistic neural units (L-units) based on bicentral functions have a natural implementation as a combination of two nodes in the MLP network [11], [29]. The basic scheme of such a unit is shown in Fig. 3. An input X is connected via W_1, W_2 weights to two neurons, each with its own separate bias b_i and b'_i. The two hidden neurons of the L-unit are connected to its output neuron via weights S_1 and S_2. All weights are exactly zero or ± 1, determining the type of the linguistic variable (Fig. 3); biases determine the position of the soft trapezoidal window. Increasing the slope of the sigmoidal function during training allows for transition from fuzzy to crisp logic. The constrained architecture MLP network (Section VI, Fig. 4) filters continuous input features through L-units and combines their logical outputs in the hidden rule units (R-units), providing logical rules. The training of L-units may alternate with the training of R-units. Training more units than needed leads to zero weights of some units, allowing for their removal.

IV. Decision Trees for Rule Generation

Decision trees are an important tool in data mining. They are fast and easy to use, although the hierarchical rules that they generate have somewhat limited power (Fig. 1). Most trees use a top-down algorithm performing a general-to-specific heuristic hill-climbing search for the best partitioning of the feature space into regions containing vectors from a single class. Heuristics are usually based on information-theoretic measures. Despite greedy hill-climbing search algorithms that may create suboptimal tree structures, in many applications, rules generated by decision trees are simple and accurate.

A. Methods Using Information Gain

The 1R algorithm leads to the simplest decision rules [32]. This algorithm searches for the best feature that has a set of values or a range of values for which vectors from a single class dominate, and presents it as a rule. ID3 [41] and its successors, C4.5 [42] and C5.0, are currently the most widely used algorithms for generating decision trees. Given a dataset $\mathcal{D}$, a decision tree is generated recursively as follows.

1) If $\mathcal{D}$ contains one or more examples, all belonging to a single class, then create a leaf node and stop.
2) If $\mathcal{D}$ contains examples belonging to a mixture of classes, information gain is used as a heuristic to split $\mathcal{D}$ into partitions (branches) based on the values of a single feature.

Suppose that each pattern in the dataset $\mathcal{D}$ belongs to one of the C_k classes, $k = 1 \ldots K$, and n_k is the number of patterns in class C_k. The amount of information contained in class distribution is

$$I(\mathcal{D}) = -\sum_{k=1}^{K} p(C_k) \log_2 p(C_k) \approx -\sum_{k=1}^{K} \frac{n_k}{N} \log_2 \frac{n_k}{N} \quad (9)$$

where the number of all patterns in the set $\mathcal{D}$ is $N = \sum_{k=1}^{K} n_k$. If the dataset is split into two subsets $\mathcal{D} = \mathcal{D}_{\mathrm{LS}} \cup \mathcal{D}_{\mathrm{RS}}$ using the split value s of feature X_i (7), the amount of information in LS and RS can be similarly computed

$$I(\mathcal{D}_{\mathrm{LS}}) = -\sum_{k=1}^{K} \frac{n_{k,\mathrm{LS}}}{N_{\mathrm{LS}}} \log_2 \frac{n_{k,\mathrm{LS}}}{N_{\mathrm{LS}}}$$

$$I(\mathcal{D}_{\mathrm{RS}}) = -\sum_{k=1}^{K} \frac{n_{k,\mathrm{RS}}}{N_{\mathrm{RS}}} \log_2 \frac{n_{k,\mathrm{RS}}}{N_{\mathrm{RS}}}$$

where $n_{k,\mathrm{LS}}$ and $n_{k,\mathrm{RS}}$ are, respectively, the number of samples in LS and RS subsets that belong to the class C_k, and N_{LS}, N_{RS} are the number of samples in LS and RS, respectively. The information gained by splitting the dataset $\mathcal{D}$ into $\mathcal{D}_{\mathrm{LS}}$ and $\mathcal{D}_{\mathrm{RS}}$ using a test on feature X_i is

$$\mathrm{IGain}\,(X_i, s) = I(\mathcal{D}) - [I(\mathcal{D}_{\mathrm{LS}}) + I(\mathcal{D}_{\mathrm{RS}})]\,. \quad (10)$$

The normalized information gain is

$$\mathrm{NGain}\,(X_i, s) = \frac{-\mathrm{IGain}\,(X_i, s)}{\sum_{j \in \{\mathrm{LS},\mathrm{RS}\}} \frac{N_j}{N} \log_2 \frac{N_j}{N}}. \quad (11)$$

A feature X_i with its corresponding value s maximizing the normalized gain is selected for splitting a node in the growing tree.

Rules obtained from the C4.5 tree are mutually exclusive, but some conditions may be spurious. The C4.5 rules algorithm [42] simplifies rule sets by deleting those conditions that do not affect accuracy significantly (the minimum description length principle is used to determine them; see [1, Ch. 7] or [2, Ch. 9]). The final set of rules may contain some rules that can be dropped, since some cases are covered by more than one rule. The remaining rules are ordered, with rules making the fewest false positive errors coming first. The majority class among those cases that are not covered by any rules becomes the default class. This leads to smaller sets of rules while preserving their accuracy.

B. Methods Using SSV

The SSV separability criterion (8) has also been used in the top-down decision tree construction algorithm [37]. Selection of the feature to be split and the split value is done using beam search, so at each stage, several (usually ten) partial trees compete until the best one emerges when the final trees are selected. If there are no contradictory examples in the data, the decision tree achieves 100% accuracy. To avoid overfitting, a cross-validation training is performed to find the optimal pruning parameters for the tree. In each cross-validation pass, the number of errors counted on the test part of the data is calculated. The degree of pruning is equal to the number of nodes removed that increase the number of errors of their parent by not more than n. The minimum total cross-validation test error (sum of all cross-validation test errors) is obtained for the optimal degree of pruning n, assuring the best generalization.

SSV decision trees have two other unique properties. The separability measure is applied not only to the individual features, but also to the linear or nonlinear combinations of the features, in effect allowing for more flexible decision borders than in typical decision trees. In particular, the test $D(\mathbf{X}, \mathbf{R}) < s$ based on distances between the data vectors $\mathbf{X}$ from the tree node and some reference vectors $\mathbf{R}$ in the feature space is used. Such tests allow for discovery of a very simple rules and useful prototypes for classes [43]. Combination of different tests leads to heterogeneous decision borders of the SSV tree and facilitates the discovery of simple class structures.

Decision tree algorithms that use the values of a single feature when splitting a node generate axis-parallel hyperplanes. If oblique hyperplanes are allowed, a more compact decision tree could be generated. Oblique decision trees are more general than univariate trees. To build an oblique decision tree, a new feature that is a linear combination of the original features is constructed at each nonleaf node. By using this new feature, the tree effectively partitions the input attribute space by hyperplanes that do not have to be axis-parallel. Several researchers [44]–[48] have proposed various approaches for computing the hyperplane weight coefficients for each node of the tree, applying such techniques as simulated annealing, randomization, regression, linear programming, and neural-inspired algorithms.

C. Methods Using Random Perturbation

CART [49] splits each node in the decision tree to maximize the purity of the resulting subsets. A node with patterns that belong only to one class has the highest purity. Nodes with patterns from several classes have a nonzero "Gini diversity index," calculated as

$$\text{Gini}(\mathcal{D}) = 1 - \sum_{k=1}^{K} \left(\frac{n_k}{N}\right)^2. \tag{12}$$

In its default setting, CART builds univariate decision trees. However, it also allows for the generation of oblique decision trees. A node split in the latter case is induced by the hyperplane $\sum_i W_i X_i - \theta \geq 0$, where X_i's are the normalized features, W_i's are the coefficients that determine the orientation of the hyperplane, and θ is a threshold. The values of W_i and θ are fine-tuned by perturbing their values to decrease the impurity of the split.

OC1 [46] combines deterministic hill climbing with randomization to find the best multivariate node split. It first finds the best axis-parallel split at a node, then looks for a better split by searching for oblique hyperplanes in the attribute space. Oblique hyperplanes with lower impurity than the best axis-parallel split are obtained by randomly perturbing the current hyperplane to a new location.

D. Hybrid Connectionist–Symbolic Method

The NN-DT algorithm [48] makes use of both connectionist and symbolic approaches to generate oblique decision trees. First, a three-layer feedforward neural network is constructed and pruned. The hidden unit activations of the pruned network are then given as input to a univariate decision-tree generating method such as C4.5. Since the hyperbolic tangent function is used to compute the activation values of the hidden units, the conditions for node splitting in the decision tree involve nonlinear terms that are the hyperbolic tangent of linear combinations of a set of input attributes. The nonlinearity, however, can be removed easily, since the hyperbolic tangent function is a one-to-one function. Thus, NN-DT effectively generates oblique decision trees.

A common problem with the decision trees is their instability, that is, sensitivity to small changes in the training data [50]. For example, quite different trees with similar accuracy can be created in a cross-validation procedure. For predictive modeling, this is treated as a disadvantage, and committees of trees are formed to stabilize the results. For discovery of interesting rules, creation of a "forest of trees" instead of a single one is actually an advantage, allowing for creation of alternative sets of rules that classify the data with similar accuracy. Robust rules that appear most frequently in many trees generated during cross validation are found in this way. Rule sets of similar overall accuracy nevertheless may differ significantly in their sensitivity and specificity. Algorithms to create forests of SSV decision trees have been presented in [43].

V. Inductive Approaches to Extraction of Logical Rules

The ML community has produced many inductive learning algorithms, also called concept learning algorithms. Many of these algorithms work only for symbolic inputs, so continuous features have to be discretized first. A comprehensive reference discussing inductive algorithms and comparing logical rules generated in this way with rules provided by other methods is not available at the time of this writing, but selected algorithms were presented in a textbook [24].

For over 30 years, Michalski has been working on the family of AQ covering algorithms [51], creating more than 20 major versions. In AQ concept description, rules for assigning cases to a class are built starting from a "seed" example selected for each class. A set of most general rules that cover this and other examples that belong to the same class is generated (this is called "a star"). An evaluation function that includes several criteria (based on the precision of the rule, the number of correctly classified examples divided by total number covered) is applied to these rules, and then the best rule is selected. Examples covered by this rule are removed from the learning set and the process is repeated. Variants include algorithms that are noise tolerant, are based on incremental learning, use many constructive induction operators to create new attributes, use evolutionary algorithms for feature selection, are hypothesis driven, and have only partial memory of presented cases (for online learning). The AQ15 program and several other algorithms were used in a multistrategy approach to data mining [52], combining ML, database, and knowledge-based technologies.

CN2 [53] is an example of a covering algorithm combining features of AQ with decision tree learning. A search for good rules proceeds in a general-to-specific order, adding new conjunctive conditions or removing disjunctive ones. It starts with a rule assigning all cases to class C: "If True Then Class=C," and performs a beam search, specializing the rule, using as search heuristics either precision (as AQ), entropy (as ID3 or C4.5), or Laplacian error estimate: $A(n, n_+, K) = (n - n_+ + K - 1)/(n + K)$, where n is the total number of examples covered by the rule, n_+ is the number of class C_+ examples, and K is the number of classes. Generated rules are either ordered (have to be used in specified sequence) or unordered.

RIPPER [54] creates conjunctive rules covering examples from a given class, adding new features in a similar way as decision trees, selecting the best subsets or splits on the basis of information gain heuristics [42]. A rule is grown on the training data (usually two-thirds of all available data) until it covers examples from a given class, and then pruned, removing the last conditions, until the difference between the number of correctly covered cases minus the incorrectly covered cases, divided by the number of all cases covered by the rule, reaches a maximum. The rule is added to the set of rules and all examples covered by it are discarded. The total number of rules is restricted by a criterion based on the minimum description length and the total set of rules optimized. On many datasets, this algorithm has found simpler and more

accurate rules than those generated by a C4.5 decision tree in the rules mode.

Version spaces (VS) is an algorithm that also belongs to the family of covering algorithms [24], [55]. The VS algorithm works with symbolic inputs, formulating hypotheses about the data $\mathcal{D}$ in the form of conjunctive rules. Such a hypothesis space $\mathcal{H}$ may be ordered according to how general or specific the hypothesis is. For example, the hypothesis $A_1 \wedge A_2 \wedge A_4$ is less general than $A_1 \wedge A_4$ or $A_1 \wedge A_2$. The version space $VS_{\mathcal{H}\mathcal{D}}$ is the subset of hypotheses from $\mathcal{H}$ consistent with all training examples in $\mathcal{D}$. The VS algorithm works by specializing general hypotheses and generalizing the specific ones.

Inductive logic programming (ILP) is a subfield of ML concerned with inducing first-order predicate calculus logic rules (FOL rules) from data (examples and additional knowledge) expressed as Prolog programs [56]. Objects classified by FOL rules may have a relational, nested structure that cannot be expressed by an attribute-value vector. This is useful for sequential data (such as those in natural language analysis or bioinformatics), and for structured data, when an unknown part of the complex object is responsible for classification (such as in chemistry).

Unfortunately, in contrast to neural networks or decision trees, software implementations of inductive ML algorithms are not readily available; therefore, it is difficult to evaluate and compare them with other methods.

VI. Neural Networks for Rule Extraction

Neural networks are regarded commonly as black boxes performing mysterious functions and representing data in an incomprehensible way. Contrary to this opinion, they can be used to provide simple and accurate sets of logical rules. Two issues should be considered: understanding what neural networks really do, and using neural networks to extract logical rules describing the data. Although the function realized by a typical neural network is difficult to understand, it may be simplified and approximated by logical rules. Many neural algorithms that extract logical rules directly from data have been devised. There are very few comparisons with other methods, and results in the form of explicit logical rules are rarely published. Several neural algorithms for rule extraction have been compared experimentally on benchmark datasets with very good results [4], [11], [57]–[60].

Compared to ML and decision tree methods, neural-inspired algorithms have important advantages, especially when the inputs are continuous. In such cases, good linguistic variables may be determined simultaneously with logical rules, and selection and aggregation of features into smaller number of useful features may be incorporated in the neural model. In addition, adaptation mechanisms for continuously changing data are built in, and wide-margin classification provided by neural networks leads to more robust logical rules.

Neural rule-extraction algorithms may be compared using six aspects (as proposed in [61], and extended in [11]): 1) the "expressive power" of the extracted rules (types of rules extracted); 2) the "quality" of the extracted rules (accuracy, fidelity comparing to the underlying network, comprehensibility and consistency of rules); 3) the "translucency" of the method, based on local–global use of the neural network (analysis of the individual nodes versus analysis of the total network function); 4) the algorithmic complexity of the method; 5) specialized network training schemes; and 6) the treatment of linguistic variables: some methods work only with binary variables, other with discretized inputs, and yet others with continuous variables that are converted to linguistic variables automatically.

A. Global Methods

In the simplest case, the inputs and outputs are binary and the network yields logical outputs. After training, the network performance is equivalent to a set of logical rules that may be found by taking all possible combinations of features as input. For n logical features X_i, the number of conjunctive rules is 3^n (in the rule antecedent, each feature may either be absent or be present as X_i or as $\neg X_i$). To limit the number of nodes in the search graph, one may try to limit the number of literals in the antecedents of extracted rules. In one of the first neural rule-extraction methods, Saito and Nakano [62] restricted the maximum number of positive and negative literals and the depth of the breadth-first search process, additionally restricting the search tree to those combinations of literals that were present in the training set. Due to these restrictions, their method sometimes creates a rule that is too general. This drawback has been removed in the method developed by Gallant [63]. The difficulty comes from the inputs that are not specified in the rule provided as a candidate by the search procedure. Gallant takes all possible values for these inputs, and although rules generated in this way are always correct, they may be too specific.

The validity interval analysis (VIA) method developed by Thrun [64] is a further extension of the global approach. A validity interval, specifying the maximum activation range for each input, may be found using linear programming techniques. These intervals may be propagated backward and forward through the network. Arbitrary linear constraints may be applied to input as well as output units, giving the method the ability to check the validity of nonstandard form of rules, such as the M-of-N rules. VIA can also handle continuous-valued input features, starting from the training values and replacing them with intervals that are increased to achieve a good generalization of the rules. The method may be applied to any neural network with monotonic transfer functions. Unfortunately, it has a tendency to extract rules that are too specific and rather numerous.

Neural network classifiers map whole feature space areas into single output numbers. If this mapping could be inverted and the input subspace leading to constant (or approximately constant) outputs characterized, logical rules could be found. Inversion techniques of feedforward neural networks have many applications (for a survey, see [65]), and have been recently applied in rule extraction using interval arithmetic [66]. The complexity of the inversion method grows exponentially with the dimension of the feature space, but in combination with feature selection techniques, this method may have some applications.

Constructive neural network with three triangular membership functions per one continuous input has recently been used to extract one dominant rule per neuron [67]. Extraction of only one dominant rule and pruning of rule conditions is computationally simple. There is no reason why good fuzzy rules should be obtained with such poor discretization, but by chance (as shown in Fig. 7 later), the Iris problem is solved quite accurately, making a false impression that it is a promising method.

B. Local Methods

Methods presented above were global, based on analysis of outputs of the whole network for various inputs. Local, or "decompositional," methods [4] analyze fragments of the network, usually single hidden nodes, to extract rules. Such networks are based either on sigmoidal functions (step functions in the logical limit) or on localized functions. Using step functions, the output of each neuron becomes logical (binary), and since the sigmoidal output functions are monotonic and their input values are between zero and one, it is enough to know the sign of the weight to determine its contribution to activation of a given unit. A search for rules involves 2^n possible combinations of input features. Rules corresponding to the whole network are combined from rules for each network node.

Local methods for extraction of conjunctive rules were proposed by Fu [68]–[71] and by Setiono and Liu [57]–[59], [72]. As with the global methods, the depth of search for good rules is restricted. The weights may be used to limit the search tree by providing the evaluation of contributions of inputs that are not specified in rule antecedents. As shown by Sethi and Yoo [73], the number of search nodes is then reduced to $O(2^n/\sqrt{n})$. In the Subset algorithm of Towell and Shavlik [74], inputs with largest weights are analyzed first, and if they are sufficient to activate the hidden node of the network irrespective of the values on other inputs, a new rule is recorded. Combinations of the two largest weights follow, until the maximum number of antecedent conditions is reached. A fuzzy version of this approach has been proposed by Hayashi [75].

C. Simplifying Rule Extraction Process

All these methods still have a problem with an exponentially growing number of possible conjunctive propositional rules. Towell and Shavlik [74] proposed to use M-of-N rules, since they are implemented in a natural way by network nodes. In some cases, such rules may be more compact and comprehensible than conjunctive rules. To avoid a combinatorial explosion of the number of possible input combinations for each network node, groups of connections with similar weights are formed. Weights in the group are replaced by their averages. Groups that do not affect the output are eliminated and biases are reoptimized for frozen weights. Such a simplified network has an effectively lower number of independent inputs; therefore, it is easier to analyze. If symbolic knowledge is used to specify initial weights, as it is done in the knowledge-based artificial neural networks (KBANN) of Towell and Shavlik [76], weights are clustered before and after training.

The search process is further simplified if the prototype weight templates (corresponding to symbolic rules) are used for comparison with the weight vectors [77]—weights are adjusted during training to make them more similar to the templates. The RuleNet method based on templates has been used to find the best M-of-N rules in $O(n^2)$ steps, and the best sets of nested M-of-N rules in $O(n^3)$ steps [78], exploring large spaces of candidate rules. The method handles only discrete-valued features; therefore, initial discretization is necessary for continuous features. The network has only one hidden layer with a specific architecture to inject symbolic rules into the network and to refine them iteratively.

Several authors noticed the need for simplifying neural networks to facilitate the rule-extraction process. Setiono and Liu [79] use a regularization term in the cost function to iteratively prune small weights. After simplification, the network is discretized by clustering activation values of the hidden unit obtained during presentation of the training set. The method does not guarantee that all rules will be found, but results for small networks were encouraging. The method of successive regularization [80] is based on a similar idea, with Laplace regularization (sum of absolute weight values) in the error function, inducing a constant decay of weights. Only weights smaller than some threshold are included in the regularizing term (this is called selective forgetting). Hidden units are forced to become fully active or completely inactive. As a result of training, only a skeletal network structure is left, and the dominant rules are extracted easily. Keeping this skeletal network frozen, small connections are revived by decreasing the regularization parameters. After training of this more complex network, additional logical rules are obtained from analysis of new nodes/connections. Another simple method belonging to this group has been presented by Geczy and Usui [81]. Weights in the MLP network with one hidden layer are mapped, after training, into 0, $+1$, or -1 values, simplifying the rule search step. In the MLP2LN approach [82], described in some detail below, such a mapping is incorporated in the learning scheme.

Rule extraction as learning (REAL) is a rather general technique introduced by Craven and Shavlik [83] for incremental generation of new rules (both conjunctive and M-of-N rules). If a new example is not classified correctly by the existing set of rules, a new rule based on the misclassified example is added, and the fidelity of the extended set of rules is checked against the neural network responses on all examples used so far. The RULENEG algorithm [4], [84], [156] is based on a similar principle: one conjunctive rule per input pattern is generated, and if a new training vector is not classified correctly by the existing set of rules $\mathcal{R}$, a new rule is created as a conjunction of all those inputs literals that have influence on the class of the vector. This is determined by consecutive negation of each input value followed by checking (using the neural network), if the predicted class has changed.

In the BRAINNE algorithm [85], a network of m inputs and n outputs is changed to a network of $m + n$ inputs and n outputs, and then retrained. Original inputs with weights that change little after extension and retraining of

the network, correspond to the most important features. The method can handle continuous inputs and has been used in several benchmark and real-life problems, producing rather complex sets of rules [4], [85]. Logical rule extraction has also been attempted using a self-organizing ART model [86] and fuzzy ARTMAP architecture [87]. In the last case, a certainty factor for each rule is provided. Simpler self-organizing architectures may also be used for rule extraction [88], although accuracy of the self-organized mapping for classification problems is rather poor [89].

The DEDEC algorithm [4], [90] extracts rules by finding a minimal information sufficient to distinguish, from the neural network point of view, between a given pattern and all other patterns. To achieve this, a new set of training patterns is generated. First, inputs are ranked in order of their importance, which is estimated by inspection of the influence of the input weights on the network outputs. Second, clusters of vectors are selected and used instead of original cases. Only those features ranked as important are used to search for conjunctive rules.

Any rule-based method may be used to approximate the neural network function on some training data. The network is used as an "oracle," providing as many training examples as needed. This approach has been used quite successfully by Craven and Shavlik in their TREPAN algorithm [91], combining decision trees with neural networks. Decision trees are induced on the training data, plus the new data obtained by perturbing the training data. The additional training data are classified by the neural network. Nodes in the decision tree are split only after a large number ($\sim 10^3$) of vectors that fall in a given node have been analyzed. Therefore, the method is more robust than direct decision tree approaches, which suffer from a small number of cases in the deeper branches. Such trees offer the best fidelity to the classification by the network. Classifiers based on ensembles of different models, similarity-based classifiers, statistical methods, or any other classifiers that produce incomprehensible models of the data may be approximated by rule-based systems in the same way.

Neural networks based on separable localized activation functions are equivalent to fuzzy logic systems [14], [92]. Each node has a direct interpretation in terms of fuzzy rules, which eliminates the need for a search process. Gaussian functions are used for inserting and extracting knowledge into the radial basis set type of networks [93]. A more general neurofuzzy system based on separable functions was proposed by Duch [23], [94]. A discussion of rule extraction using localized transfer functions has been given by Andrews and Geva [95], [96]. These authors developed a quite successful approach called RULEX [97], based on constrained MLP networks with pairs of sigmoidal functions combined to form "ridges," or "local bumps." Rules in this case are extracted directly from an analysis of the weights and thresholds, since disjointed regions of the data are covered by one hidden unit. In effect, the method is similar to a localized network with steplike activation functions. The method works for continuous as well as discrete inputs.

Methods of combining neural and symbolic knowledge, refining probabilistic rule bases, scientific law discovery [98], and data mining [30], are related closely to the applications of neural networks for extraction of logical rules. Symbolic rules may be converted into RAPTURE networks [99] and trained using a modified backpropagation algorithm for optimization of certainty factors. The network prunes small connections and grows by adding new nodes if classification accuracy becomes too low.

It may seem that neurofuzzy systems should have advantages in application to rule extraction, since crisp rules are just a special case of fuzzy rules. Many neurofuzzy systems have been constructed [23], [100]–[103]. However, there is a danger of overparametrization of such systems, leading to difficulty of finding optimal solutions even with the help of evolutionary algorithms or other global optimization methods [104]. Systems based on rough sets [19] require additional discretization procedures that may determine the quality of their performance. We have included a few results obtained by fuzzy and rough systems in Section IX. Algorithms deriving very simple crisp logic rules, based on decision trees or neural networks, may have advantages over the fuzzy, rough, or neurofuzzy systems. Unfortunately, many rule-extraction methods have been tested on datasets that are not in the public domain; therefore, their relative advantages are hard to assess.

D. The MLP2LN Algorithm

To facilitate extraction of logical rules from an MLP network, one could transform it smoothly into a network performing logical operations [a logical network (LN)]. This transformation is the basis of the MLP2LN algorithm [105]. Skeletonization of a large MLP network is the method of choice if the goal is to find logical rules for an already-trained network. Otherwise, starting from a single neuron and constructing the LN using training data directly (constructive, or C-MLP2LN algorithm) is faster and usually more accurate. Since interpretation of the activation of the MLP network nodes is not easy [106], a smooth transition from MLP to a logical type of network performing similar functions is advocated. This transition is achieved during network training by the following.

1) Increasing gradually the slope β of sigmoidal functions $\sigma(\beta x)$ to obtain crisp decision regions.
2) Simplifying the network structure by inducing the weight decay through a penalty term.
3) 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 error function

$$E(W) = \frac{1}{2}\sum_{\mathbf{X}} \|C(\mathbf{X}) - M(\mathbf{X};W)\|^2 + \frac{\lambda_1}{2}\sum_{i,j} W_{\mathrm{ij}}^2 + \frac{\lambda_2}{2}\sum_{i,j} W_{\mathrm{ij}}^2 (W_{\mathrm{ij}}-1)^2 (W_{\mathrm{ij}}+1)^2. \quad (13)$$

The first part is the standard mean-square-error (MSE) measure of matching the network output $M(\mathbf{X};W)$ with the desired output class $C(\mathbf{X})$ for all training data samples

$\mathbf{X}$. The second term, scaled by λ_1, is used frequently in the weight pruning or in the Bayesian regularization method [107], [108] to improve generalization of the MLP networks.

A naive interpretation of why such regularization works is based on the observation that small weights and thresholds mean that only the linear part of the sigmoid around $\sigma(0)$ is used. Therefore, the decision borders are rather smooth. On the other hand, for logical rules, sharp decision borders are needed and a simple skeletal network is necessary. To achieve these objectives, the first regularization term is used at the beginning of the training to force some weights to become sufficiently small to remove them.

The second regularization term, scaled by λ_2, is a sum over all weights and has minimum (zero) for weights approaching 0 or ± 1. The first term is switched off and the second increased in the second stage of the training. This allows the network to increase the remaining weights and, together with increasing slopes of the sigmoidal functions, to provide sharp decision borders.

In the error backpropagation training, partial derivatives $\partial E(W)/\partial W_{\mathrm{ij}}$ of the error function (13) are calculated. The two additional terms in the error function contribute

$$\lambda_1 W_{\mathrm{ij}} + \lambda_2 W_{\mathrm{ij}} \left(W_{\mathrm{ij}}^2 - 1\right) \left(3W_{\mathrm{ij}}^2 - 1\right) \qquad (14)$$

to this derivative. Although nonzero weights have values restricted to ± 1, increasing the slopes β is equivalent to using a single, large nonzero weight value $\pm W$.

R-units combine the logical inputs received from L-units that are responsible for creating the hyperrectangular subspaces. Domain knowledge that may help to solve the problem may be inserted directly into the network structure, defining initial conditions which could be modified further in view of the incoming data. Since the final network structure becomes quite simple, inserting partially correct rules to be refined by the learning process is quite straightforward.

The training may proceed separately for each output class. A constructive procedure frequently leads to satisfactory solutions after a very fast training. A single hidden neuron (R-unit neuron) per class is created and trained using a backpropagation procedure with regularization. λ_1 and the slope of sigmoidal function β are increased gradually and weights with magnitude smaller than 0.1 are removed. λ_2 is then increased until the remaining weights reach 0 ± 0.05 or $\pm 1 \pm 0.05$. Finally, very large slopes $\beta \approx 1000$ and integer weights $0, \pm 1$ are set, effectively converting neurons into threshold logic functions. The weights of existing neurons are frozen and new neurons (one per class) are added and trained on the remaining data in the same way as the first one. This procedure is repeated until all data samples are classified correctly, or until the number of rules obtained grows sharply, signifying overfitting (for example, one or more rules per one new vector classified correctly are obtained).

The C-MLP2LN network expands after a neuron is added and then shrinks after connections with small weights are removed. A set of rules $\mathcal{R}_1 \vee \mathcal{R}_2 \ldots \vee \mathcal{R}_n$ is found for each class separately. 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 (Fig. 4). Output neurons performing summation of the incoming signals are linear and have either positive weight $+1$ (adding more rules) or negative weight -1. The last case corresponds to those rules that cancel some of the errors created by the rules found previously that were too general. They may be regarded as exceptions to the rules.

Since only one neuron per class is trained each time, the C-MLP2LN training is fast. Both a standard MLP architecture with linguistic inputs or the L–R network may be used with the C-MLP2LN approach. Since the first neuron for a given class is trained on all data for that class, the rules it learns are most general, covering the largest number of instances. Therefore, rules obtained by this algorithm are ordered, starting with rules that have the largest coverage and ending with rules that handle only a few cases. This ordering allows for a very easy check of the quality of a set of rules by looking at the errors on the training data. An optimal balance between the number of rules and the generalization error is usually obtained when only the rules that cover a larger number of cases are retained.

The final solution may be presented as a set of rules, or as a network of nodes performing logical functions, with hidden neurons realizing the rules, and the hidden-output neuron weights set to ± 1. However, some rules obtained from analysis of the network may involve spurious conditions; therefore, an optimization and simplification step is necessary (see Section VII).

Although constraints (13) do not transform the MLP exactly into an LN, after they are enforced successfully, they are sufficient to facilitate logical interpretation of the final network function (Fig. 5). The λ_1 and λ_2 parameters determine the simplicity/accuracy tradeoff of the generated network and extracted rules. If a very simple network (and, thus, simple logical rules) is desired, providing 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 MSE, a few experiments are sufficient to find the largest value for which the MSE is still acceptable, and does not decrease sharply when λ_1 is decreased. Smaller values of λ_1 should be used to obtain more accurate networks (larger sets of rules). The final value of λ_2 near the end of the training may grow larger than the maximum value of λ_1.

An MLP network is transformed into an LN by increasing the slope of sigmoidal functions to infinity, changing them into the step functions. Such a process is difficult, since a very steep sigmoidal function leads to the nonzero gradients only in small regions of the feature space; thus, the number of vectors contributing to the learning process goes to zero. Therefore, when convergence becomes slow for large slopes, it is necessary to stop network training, extract logical rules, and optimize the intervals of the linguistic variables. This optimization step, described in Section VII, is performed at the level of the rule-based classifier, not the MLP network.

E. Search-Based Procedures

The search-based training procedure is an interesting, although seldom used, alternative to the gradient-based back-

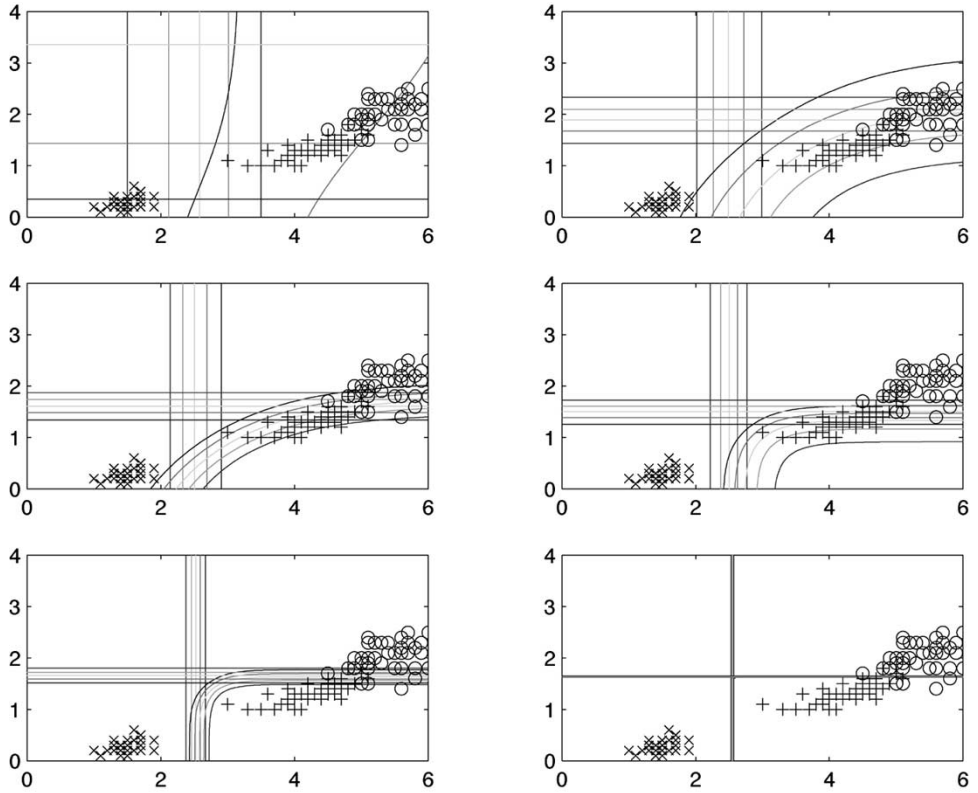

Fig. 5. Convergence of the MLP2LN network for the Iris data. Five contours of constant values (0.3–0.7, with 0.5 in the middle) for three hidden sigmoidal functions are shown. The data is displayed in x_3 (petal length) and x_4 (petal width) coordinates. Small sigmoidal slopes (top left drawing) at the beginning of the training gradually increase, becomeing stepwise (lower right), with contours collapsing to a single line. The space is now partitioned into hyperboxes.

propagation training [109]–[111]. Quantization of network parameters (weights and biases) allows for replacement of gradient minimization by a search procedure. Steplike discontinuous transfer functions, as well as any smooth functions, may be used in search-based optimization. Increasing step by step the resolution of quantization from coarse to fine, a search for the optimal network parameters is made with arbitrary precision. Replacing the gradient-based backpropagation training methods by global search algorithms to minimize the value of the error function is rather expensive; therefore, some form of a heuristic search should be used, for example, best-first search or beam search [112]. Even if the best-first search algorithm is used (corresponding to the steepest gradient descent), a good solution may be found by gradually increasing the resolution of the discrete network parameters [109]. In backpropagation training, this would correspond roughly to a period of learning with rather large learning constants, with some annealing schedule for decreasing the learning constant.

The algorithm starts with one neuron per class and all weights $W_{\mathrm{ij}} = 0$ and biases $\theta_i = -0.5$, so that all data are assigned to the default class (corresponding to zero network output). At the beginning of the search procedure, the step value Δ for weights (and biases) is set. This value is added or subtracted from weights and biases $W_{\mathrm{ij}} \pm \Delta$, $\theta_i \pm \Delta$. Best-first search or beam search strategies are used to modify one or two network parameters at a time. To speed up the search, all single changes of parameters are tested and a number of the most promising changes (i.e., changes decreasing the value of the cost function) are selected (the beam width). Second, all pairs of parameter changes from the chosen set, or even all the subsets of this set, are tested, and the best combination of changes applied to the network. Since the first stage reduces the number of weights and biases that are good candidates for updating, the whole procedure is computationally efficient. The value of Δ is reduced in half if no further improvement is found. After training of the initial neurons, more neurons are added to the classes that have not been separated correctly, and the search procedure is repeated for the new weights and biases only.

VII. Optimization, Reliability, and the Use of Sets of Rules

Controlling the tradeoff between comprehensibility and accuracy, optimizing the linguistic variables and final rules, and estimating the reliability of rules are rarely discussed in papers on logical rule extraction. In practical applications, it may be quite useful to have a rough, low-accuracy, simple description of the data and to be able to provide a more accurate but more complex description in a controlled manner. Neural or other methods of logical rule extraction may provide initial rules, but their further refinement is often desirable [113].

Rules obtained from analyzing neural networks or decision trees 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, an important part of rule optimization involves simplification and symbolic operations on rules. A Prolog program may be used for such simplifications [114]. In addition, optimal linguistic variables for continuous-valued features may be found for the sets of rules extracted. These optimized linguistic variables may be used to extract better rules in an iterative process, starting from initial values of linguistic variables, extracting logical rules, optimizing linguistic variables, and repeating the whole process with new linguistic variables until convergence is achieved. Usually two or three iterations are sufficient to stabilize the sets of rules.

A. What to Optimize?

Optimal linguistic variables (intervals) and other adaptive parameters may be found by maximization of predictive power of a rule-based (or any other) classifier. Confidence in the classification model may be improved at a cost of rejecting some cases, i.e., assigning them to the "unknown" class. After training a neural network or other classifiers, outputs from these models may be interpreted as probabilities. Suppose that the confusion matrix $\mathcal{F}(C_i, C_j|M)$ for a two-class problem is known

$$\mathcal{F}(C_i, C_j|M) = \frac{1}{n}\begin{pmatrix} n_{++} & n_{+-} & n_{+r} \\ n_{-+} & n_{--} & n_{-r} \end{pmatrix} \tag{15}$$

with rows corresponding to true classes, and columns to the predicted classes. Thus, n_{++} vectors from the true class C_+ are correctly recognized as C_+ [true positives (TP)], and n_{+-} vectors from the class C_+ are incorrectly assigned to the class C_- [false negatives (FN)]; n_{+r} is the number of C_+ class vectors that have been rejected by the model as "unknown" (if there is no default class covered by the "else" condition, rule conditions may not cover the whole feature space). In the second row, analogous quantities for the C_- class are stored, n_{--} [true negatives (TN)], n_{-+} [false positives (FP)], and n_{-r}, unclassified vectors from the C_- class. The sum of first-row entries $n_+ = n_{++}+n_{+-}+n_{+r}$ is equal to the total number of the C_+ vectors and of the second-row entries n_- to the number of C_- vectors.

Probabilities $p_{ij} = n_{ij}/n$ are computed for a model M on $n = n_+ + n_-$ samples. The $p_{ij} = p(C_i, C_j|M)$ quantities are the training set estimations of probabilities of predicting class C_j if the true class is C_i; the model may also reject some cases as unpredictable, assigning them to the class C_r. The $p_\pm = p(C_\pm) = n_\pm/n$ are the *a priori* probabilities for the two classes $p_\pm = p_{\pm+} + p_{\pm-} + p_{\pm r}$ and $p_+ + p_- = 1$. They should not be confused with $p_\pm(M) = p_{+\pm}(M) + p_{-\pm}(M)$ or probabilities that model M will assign a vector to class $C_\pm$.

To increase confidence in the decision of the model, the errors on the test set may be decreased at the cost of rejecting some vectors. In neural networks, this is done by defining minimum and maximum thresholds for the activity of output units. Similar thresholds may be introduced in models estimating probability. The following error function may be used for corrections after training of the model to set these thresholds:

$$E(M;\gamma) = \gamma \sum_{i\neq j} \mathcal{F}(C_i, C_j|M) - \mathrm{Tr}\,\mathcal{F}(C_i, C_j|M) \geq -1. \tag{16}$$

If the parameter $\gamma \geq 0$ is large, the number of errors after minimization may become zero, but some input vectors will not be recognized (i.e., rules will not cover the whole input space). Optimization of the cost function $E(M;\gamma)$ allows one to explore the **accuracy–rejection rate tradeoff**. This function is bounded by -1 and should be minimized over parameters in M without constraints. For discontinuous cost function $E(M)$ (crisp logic rules), this minimization may be performed using simulated annealing or multisimplex global minimization methods.

For a single rule R, one may minimize

$$E(R;\gamma) = \frac{(\gamma p_{-+}(R) - p_{++}(R))}{p_+(R)} \tag{17}$$

i.e., the difference between the probability of FPs minus TPs, divided by the rule coverage for the C_+ class (some algorithms minimize only $\gamma = 1$ case [54]). Several other quantities are used to evaluate the quality of classification models M, as follows:

- overall accuracy $A(M) = p_{++}(M) + p_{--}(M)$;
- overall error rate $L(M) = p_{-+}(M) + p_{+-}(M)$;
- overall rejection rate $R(M) = p_{+r}(M) + p_{-r}(M) = 1 - L(M) - A(M)$;
- sensitivity $S_+(M) = p_{+|+}(M) = p_{++}(M)/p_+$, or conditional probability of predicting class C_+ when the vector was indeed from this class;
- specificity $S_-(M) = p_{-|-}(M) = p_{--}(M)/p_-$ (same as above, but for class C_-);
- precision $PR(M) = p_{++}(M)/p_+(M)$.

Sensitivity is also called "recall" or "detection rate," since it is the proportion of correctly recognized cases C_+, while (1-Specificity) is the false alarm rate $p_{+-}(M)/p_- = FN/n_-$. Note that the overall accuracy is equal to a combination of sensitivity and specificity weighted by the *a priori* probabilities

$$A(M) = p_+ S_+(M) + p_- S_-(M). \tag{18}$$

Thus, sensitivity (specificity) plays the role of accuracy of the model for the $C_+(C_-)$ class only, with $p_+(p_-)$ being the fraction of samples from this class (all other classes) in the training set. In the K-class problem, one can always use a separate model to distinguish a single class C_+ from all other classes C_-. The cost function for the model M_+ is (all $p_{ij} = p_{ij}(M_+)$)

$$\begin{aligned} E(M_+;\gamma) &= \gamma L(M_+) - A(M_+) \\ &= \gamma(p_{+-} + p_{-+}) - (p_{++} + p_{--}). \end{aligned} \tag{19}$$

This function should be minimized over parameters of the M_+ model created for the C_+ class. For large γ, only the error is important and it may decrease at the expense of the

rejection rate—for example, by making the rule intervals tighter or the thresholds for neural activity closer to one and zero. In an extreme case, no errors will be made on the training set, since the classifier will reject all such cases. For $\gamma = 0$, only accuracy is maximized, leading to fewer rejections. Using the error (loss) and the rejection rate, (19) becomes

$$\min_M E(M;\gamma) \Leftrightarrow \min_M \{(1+\gamma)L(M) + R(M)\}. \quad (20)$$

For $\gamma = 0$, a sum of the overall error and rejection rate is minimized, while for large γ, the error term dominates, allowing the rejection rate to grow. In many applications, it is important to achieve the highest sensitivity or specificity. The error function (20) distinguishes only one of these quantities. Relative costs allow for selective optimization of sensitivity and specificity. If the cost of assigning vectors from true class C_- to the predicted class C_+ is set to one, and the cost of making an opposite error is α, the cost function (20) becomes

$$\min_M E(M;\alpha) = \min_M \{p_{+-}(M) + \alpha p_{-+}(M)\} \Leftrightarrow$$
$$\max_M \{p_+ S_+(M) + \alpha p_- S_-(M) + p_{+r}(M) + \alpha p_{-r}(M)\}. \quad (21)$$

For $\alpha = 0$, this is equivalent to the maximization of $p_{++}(M) + p_{+r}(M)$, and for large α, to the maximization of $p_{--}(M) + p_{-r}(M)$.

Receiver operator characteristic (ROC) curves show the tradeoff between sensitivity S_+ (detection, or TP rate) and $1 - S_-$ (FP, or alarm, rate), allowing for another way of adjusting the rejection thresholds. If a classifier M produces probabilities $p(C_+|\mathbf{X}; M)$, a threshold θ may be set such that vector $\mathbf{X}$ is assigned to the class C_+ if $p(C_+|\mathbf{X}; M) \leq \theta$. Each value of θ corresponds to the point $S_+(\theta)$ and $1 - S_-(\theta)$ on the ROC curve. For $\theta = 0$, both $S_+(0) = 0$ and $1 - S_-(0) = 0$, while for $\theta = 1$, both $S_+(0) = 1$ and $1 - S_-(0) = 1$. Random guessing should produce the same proportion of TPs to FPs, taking account of the *a priori* proportions found in the data, i.e., n_+/n_-. This corresponds to a point on the diagonal of the ROC curve, since in this case $S_+(\theta) = 1 - S_-(\theta)$. A crisp rule classifier produces only one point $P = (S_+, 1 - S_-)$ (in Fig. 6 points for two classifiers, A and B, are shown). The area under the line connecting (0,0) with point P plus the line connecting P with (1,1), is known as the area under ROC curve (AUC) [115]. For a crisp rule classifier AUC $= (S_+ + S_-)/2$. Thus, different combinations of sensitivity and specificity give the same AUC as long as the sum $S_+ + S_-$ is constant. Maximization of AUC is equivalent to the minimal cost solution ((21)) with no rejections and $\alpha = p_+/p_-$.

Fuzzy rules may in principle achieve higher accuracy and better ROC curves. Unfortunately, typical algorithms for fuzzy rule construction or extraction [13], [100], [116] generate a large number of such rules, so the issues of overfitting, generalization, and comprehensibility become very important in this case. There are many ways to optimize fuzzy rules. One idea is to select the most important membership function and update their parameters using gradient-based cost function minimization [117]. Genetic algorithms are quite frequently used for optimization of

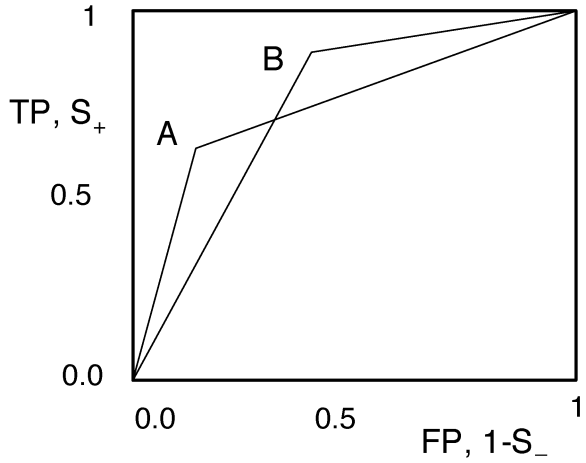

Fig. 6. ROC curves for two crisp logic rule-based classifiers, A and B, with identical area under the curve. ROC curves are usually presented using the (FP, TP) axes or, equivalently, $1 - S_-$ and S_+ axis.

membership funcitons as well as the number of rules. Entropy-based genetic optimization [104] seems to give quite good results.

B. How Reliable Are Optimized Rules?

Estimating the reliability of the rules is very important in many applications. Tests of classification accuracy should be performed using stratified cross validation, each time including rule optimization on the training set. A hierarchy of rule sets may be created by changing the value of γ in (20) to obtain models with increasing classification accuracy at the expense of larger rejection rate. A set of rules that classifies selected training cases 100% correctly for all data partitionings may be used with high confidence in its prediction; cases that are not covered by this set of rules require another set of rules of lower accuracy (the accuracy is estimated on the training set by cross validation).

Logical rules, like many 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 (as shown in Fig. 9 later). 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. Using the first method, one determines the largest cuboid (in the parameter space) in which the number of errors is constant, starting from the values of the optimized parameters. The center of this cuboid is taken as the final estimation of the adaptive parameters. The second method is equivalent to a specific form of regularization of the classification model [108]. A better method to overcome the brittleness problem based on uncertainty of inputs is described below.

Neural systems have good generalization properties because they are wide-margin classifiers. Their decision borders result from the MSE optimization of a smooth function that covers large neighborhood contributing to the error. This leads to three important features: 1) rules are extracted from data using inexpensive gradient methods instead of global

minimization; 2) more robust rules with wider classification margins are found; 3) class probability, instead of crisp 0–1 decisions, may be estimated.

Because gradient methods suffer from the local minima problem, a good initialization, or repeated starts from random initialization, should be used.

C. Probabilities From Rules

Input values are usually taken from observations that may not be completely accurate. Therefore, instead of the attribute value X, a Gaussian distribution $G_X = G(Y; X, S_X)$ centered at X with dispersion S_X should be given. These dispersions are set initially using the estimation of uncertainties of a particular type of features, to be optimized later (see below). This distribution may be treated as a fuzzy number with G_X membership function. A Monte Carlo procedure may be performed to compute probabilities $p(C_k|\mathbf{X})$: vectors $\mathbf{X}$ are sampled from a Gaussian distributions of all attributes, and results of classification are averaged to obtain probabilities. For crisp rules, analytical evaluation of this probability is based on the cumulative distribution function [11]

$$\begin{aligned} p(X \le a) &= \int_{-\infty}^{a} G(Y; X, S_X)dY \\ &= \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{a-X}{S_X\sqrt{2}}\right)\right] \\ &\approx \sigma\left(\beta(a-x)\right) \end{aligned} \tag{22}$$

where erf is the error function and $\beta = 2.4/\sqrt{2}S_X$ makes the erf function similar to the standard unipolar sigmoidal function with the accuracy better than 2% for all X. Taking a logistic function instead of the erf function corresponds to an assumption about the error distribution of X being not a Gaussian, but rather $\sigma(X)(1-\sigma(X))$. This distribution is very similar to the Gaussian distribution. For unit slope of the logistic function and $S_X^2 = 1.7$, these two distributions differ at each point by less than 3.5%. If the rule $R_{a,b}$ involves closed interval $[a, b]$, the probability that fuzzy Gaussian variable X is in this interval

$$p\left(R_{a,b}(G_X)=T\right) \approx \sigma\left(\beta(X-a)\right) - \sigma\left(\beta(X-b)\right). \tag{23}$$

Thus, the probability that a given condition is fulfilled is proportional to the value of the soft trapezoid function realized by L-unit. Crisp logical rules with the assumption that data have been measured with finite precision lead to soft L-functions that allow for calculation of classification probabilities that are no longer binary. ROC curves become smooth and the area under them (AUC) is larger than that for crisp sets of rules. Crisp logical rules obtained from any algorithm may be fuzzified or obtained directly from neural networks. Crisp logical rules with the assumption of input uncertainties are equivalent to fuzzy rules with specific membership functions. The ease of interpretation favors crisp rules, while the accuracy and the possibility of application of gradient-based techniques for optimization favors fuzzy rules.

The probability that a vector $\mathbf{X}$ satisfies a rule $\mathcal{R} = r_1 \wedge \ldots \wedge r_N$ may be defined as the product of probabilities for all rule conditions. Such definition assumes that all the attributes that occur in the rule $\mathcal{R}$ are mutually independent, which is usually not the case. If a pair of strongly dependent attributes is used in linguistic variables of a single rule, one of these variables may be dropped and the other reoptimized at the stage of rule simplification. Therefore, the product should be a good approximation to real probability. Obviously, the rule may not contain more than one premise per attribute, but it is easy to change the rules appropriately if they do not satisfy this condition.

Another problem occurs when the vector $\mathbf{X}$ belongs to a class covered by more than one rule. Rules may overlap because they use only a subset of all attributes, and antecedents of different rules are not mutually exclusive. Summing and normalizing probabilities obtained for different classes may give results quite different from real Monte Carlo probabilities. To avoid this problem, probabilities from single rules are added and then joint probabilities subtracted. For two rules $R_1(X)$, $R_2(X)$ for class C, the probability $p(C|\mathbf{X}; M)$ is $p(\mathbf{X} \in R_1) + p(\mathbf{X} \in R_2) - p(\mathbf{X} \in R_1 \cap R_2)$. For more rules, care should be taken to correctly add and subtract all terms.

An error function based on the number of misclassifications is usually used to optimize rule parameters. The error function may also be based on the sum over all probabilities

$$E(M, S_{\mathbf{X}}) = \frac{1}{2}\sum_{\mathbf{X}}\sum_{i}\left(p(C_i|\mathbf{X}; M) - \delta\left(C(\mathbf{X}), C_i\right)\right)^2 \tag{24}$$

where M includes intervals defining the linguistic variables, $S_{\mathbf{X}}$ are Gaussian uncertainties of all inputs, and $p(C_i|\mathbf{X}; M)$ is calculated using (23). The confusion matrix computed using probabilities instead of the error counts allows for optimization of (16) using gradient-based methods.

Uncertainties $S_{\mathbf{X}}$ of the values of features are additional adaptive parameters that may be optimized. In the simplest case, all S_{X_i} are taken as a percentage of the range of each feature $S_{X_i} = S(X_{i,\max} - X_{i,\min})$, and one-dimensional (1-D) minimization of the error function over S is performed. Minimization of the soft error function may lead to the increase of the number of misclassifications (for vectors near decision borders with similar probabilities of different classes), despite the fact that the overall probability of correct classification will increase. A few iterative steps should lower the number of misclassifications. After each minimization, S is decreased and minimization repeated; for sufficiently small S, probabilities are almost binary. In the limit, minimization of the soft error function (24) becomes equivalent to minimization of the number of misclassifications, but the brittleness problem is solved because the intervals that are placed optimally for larger input uncertainties do not change in subsequent minimizations.

VIII. Extraction of Rules—Example

The process of rule extraction is illustrated here using the well-known Iris dataset, provided by Fisher in 1936. The data have been obtained from the UCI ML repository [118]. The Iris data have 150 vectors evenly distributed in three classes:

iris-setosa, iris-versicolor, and iris-virginica. Each vector has four features: sepal length X_1 and width X_2, and petal length X_3 and width X_4 (all given in centimeters).

A. Data Preparation

The simplest way to obtain linguistic variables, often used in design of fuzzy systems, is based on division of each feature data range into a fixed number of parts and use of the triangular (or similar) membership functions for each part [13], [14]. The same approach may be used for crisp logic. Dividing the range of each feature into three equal parts, called small (s), medium (m), and large (l) the $X_1 \in [4.3, 7.9]$ feature will be called small if $X_1 \leq 5.5$, medium if $X_1 \in (5.5, 6.7]$, and large if $X_1 > 6.7$. Instead of four continuous-valued inputs, a network with 12 bipolar inputs equal to ± 1 is constructed. An input vector may now be written in symbolic form as (s_1, m_2, l_3, l_4), written for simplicity as (s, m, l, l). If the value of X_k is small, the network input should be $s_k = +1$, $m_k = -1$ and $l_k = -1$.

With this discretization of the input features, three vectors of the iris-versicolor class ([coded as (m, m, l, l), (m, l, m, l), and (m, s, l, m)] become identical with some iris-virginica vectors and cannot be classified correctly, yielding the maximum classification accuracy of 98%. Such vectors should be removed from the training sequence.

The accuracy of classification using logical rules depends critically on selection of linguistic variables. Although there is no reason why dividing feature ranges into a fixed number of intervals should provide good linguistic units, for the Iris example, by chance, it is quite good. Using two variables per feature, small and large, that divide the range of feature values in the middle, 13 vectors from the iris-setosa class become identical to some vectors from the two other classes. Using four linguistic variables per feature also decreases classification accuracy, making 16 iris-versicolor cases identical to iris-virginica. Evidently, division into three classes is a fortuitous choice. Analysis of the histograms of the individual features for each class, shown in Fig. 7 and Table 1, proves that the division into three equal parts is almost optimal, cutting the histograms into the regions where vectors from a single class dominate. For example, the iris-virginica class is more frequent for the value of X_3 above 4.93 and iris-versicolor are more frequent below this value. Discretization based on histograms (shown in Table 1) is certainly better. The data range was divided into 15 bins and histograms were smoothed by counting not only the number of vectors falling in a given bin, but also by adding 0.4 to adjacent bins.

B. Rule Extraction

The rule extraction process is illustrated below using the constructive C-MLP2LN method [11]. The histogram-based discretization is quite useful for the initialization of L-units, although random initialization would, after some training, also lead to similar intervals. A single neuron per class is sufficient to train the C-MLP2LN network to maximum accuracy (three errors, due to the discretization). The final network structure (Fig. 8) has 12 input and 3 output nodes.

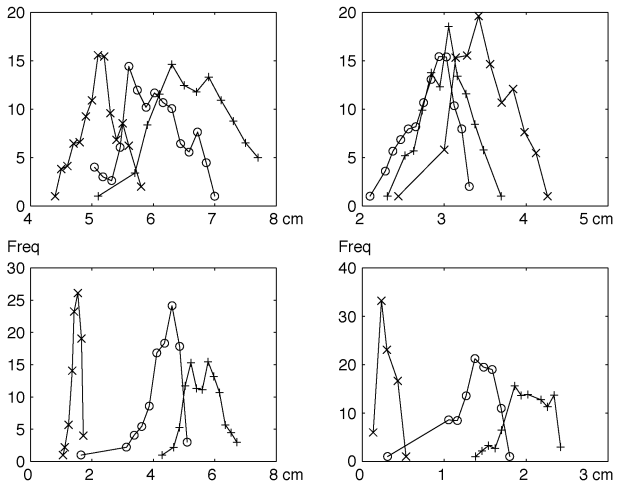

Fig. 7. Histograms of the four $X_1 - X_4$ features of iris flowers. Thirty bins were used and simple smoothing applied. The horizontal axis is length in centimeters; the vertical axis is the number of samples in a given bin. The X_1 and X_2 histograms (upper row, left and right) overlap strongly, but the X_3 and X_4 features (lower row, left and right) allow for better discrimination than the first two features.

Table 1
Linguistic Variables Obtained by Analysis of Histograms

	s	m	l
X_1	[4.3,5.5]	(5.5,6.1]	(6.1,7.9]
X_2	[2.0,2,75]	(2.75,3.2]	(3.2,4.4]
X_3	[1.0,2.0]	(2.0,4.93]	(4.93,6.9]
X_4	[0.1,0.6]	(0.6,1.7]	(1.7,2.5]

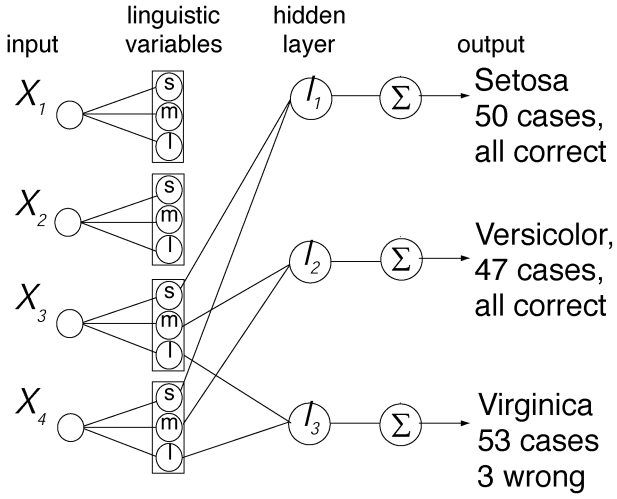

Fig. 8. Final structure of the network for the Iris problem.

Nodes representing linguistic variables in Fig. 8 are constructed from L-units (Fig. 3). The hidden-layer nodes represent rules (R-nodes), and are identical with the output layer, simplifying the general structure of the L–R network shown in Fig. 4. Separate hidden nodes are needed only when more than one neuron is necessary to represent the rules for a given class. The network was trained with the backpropagation algorithm for about 1000 epochs, and the final weights were within 0.05 from the desired ± 1 or 0 values. After training, the slopes of sigmoidal functions are set to a very large value,

changing the L-unit functions $L(X_i)$ into rectangular filters (Fig. 3), and the R-units into threshold logic functions $\Theta(\mathbf{W} \cdot \mathbf{L}(\mathbf{X}) - \theta)$. The following weights and thresholds for the three neurons were obtained (only the signs of the weights are written):

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

These weight vectors are so simple that all input combinations leading to activations above the threshold are noticed immediately. The corresponding rules are

$$\text{iris} - \text{setosa if } X_3 = s \vee X_4 = s$$
$$\text{iris} - \text{versicolor if } X_3 = m \wedge X_4 = m$$
$$\text{iris} - \text{virginica if } X_3 = l \vee X_4 = l.$$

From the trained network shown in Fig. 8, it is clear that only two features, X_3 and X_4, are relevant, since all weights for the remaining features become zero. The first rule correctly classifies all samples from the iris-setosa class. Together with the other two rules, 147 vectors (98%) are classified correctly using only the X_3 and X_4 features.

C. Alternative Rule Sets

In contrast to classification systems that are better if they have lower cross-validation errors, rule-extraction algorithms should not give just one unique answer. Using the L–R network, several solutions may be found, depending on the regularization parameters λ. In the rules presented above, linguistic variables were derived by inspection of the histograms and were not optimized. As a result, the solution obtained is rather brittle (Fig. 9), since the decision borders are placed too close to the data.

The simplest set of rules $\mathcal{R}^{(1)}$ with optimized linguistic variables involve only one attribute, petal length X_3

$$\mathcal{R}_1^{(1)}: \text{iris} - \text{setosa if } X_3 < 2.5 \quad (100\%)$$
$$\mathcal{R}_2^{(1)}: \text{iris} - \text{virginica if } X_3 > 4.8 \quad (92\%)$$
$$\mathcal{R}_3^{(1)}: \text{ELSE iris} - \text{versicolor} \quad (94\%).$$

The first rule is accurate in 100% of the cases, since it easily separates the setosa class from the two other classes. The overall accuracy of these three rules is 95.3% (seven errors). Slightly more accurate rules (96%) are obtained with smaller regularization parameters

$$\text{iris} - \text{setosa if } X_3 \leq 2.56$$
$$\text{iris} - \text{virginica if } X_4 > 1.63$$
$$\text{iris} - \text{versicolor otherwise.}$$

All these rules are more robust than those obtained with linguistic variables from histograms. More complex solutions are found using $\lambda_1 = 0$ and a small value of λ_2. To find all rules that are compatible with a given set of weights and thresholds, one has to perform a search process, considering combinations of all inputs to the activation of the network node. Because of regularization, only relevant inputs have nonzero weights; therefore, the search space has 2^n elements, where the number of used features n is usually much smaller than the number of all features. An efficient method to perform such analysis has been presented in [11]. The additional complexity of new rules should be justified by the increase of their accuracy. For the Iris data with histogram discretization, the highest accuracy of rules is 98% and has already been achieved with the rules given above.

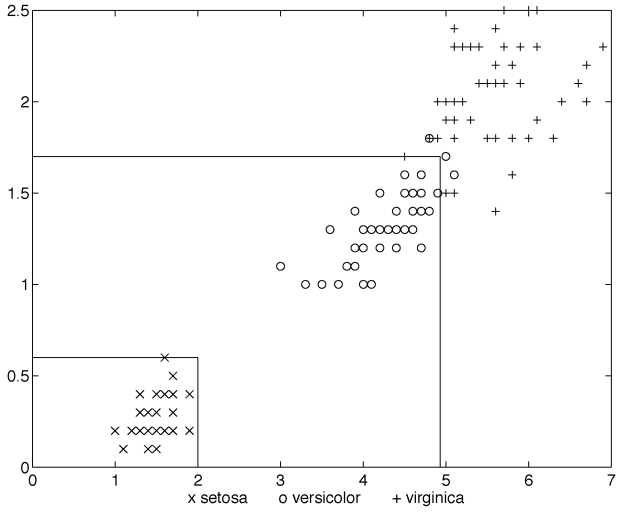

Fig. 9. Iris dataset displayed in the petal length X_3 (horizontal) and width X_4 (vertical) coordinates; decision regions (rules) for the three classes are also shown. Note the three iris-versicolor cases that are incorrectly classified using these two features. The brittleness of rules is illustrated by the decision border placed too close to the vectors from the setosa class.

The cost function (16) allows for final optimization of linguistic variables. Different values of parameters γ and λ_1 (λ_2 is not so important here) lead to a hierarchy of rules with increasing reliability. The simplest set of rules $\mathcal{R}^{(1)}$ has used only one feature X_3. Lowering the final hyperparameter λ_1 and optimizing the intervals using L-units leads to the following set of rules:

$$\mathcal{R}_1^{(2)}: \text{setosa if } (X_3 < 2.9 \vee X_4 < 0.9) \quad (100\%)$$
$$\mathcal{R}_2^{(2)}: \text{versicolor if } (X_3 \in [2.9, 4.95] \wedge X_4 \in [0.9, 1.65]) \quad (100\%)$$
$$\mathcal{R}_3^{(2)}: \text{virginica if } (X_3 > 4.95) \vee (X_4 > 1.65) \quad (94\%).$$

The $\mathcal{R}^{(2)}$ set of rules correctly classifies 147 vectors, achieving the overall 98.0% accuracy. Since linguistic variables have been optimized, there is no theoretical accuracy limit due to the discretization. Still, due to overlapping distributions of data from different classes, it may not be possible to remove all errors. The first two rules have 100% reliability, while all errors are due to the third rule, covering 53 cases. With further decrease of the constraint hyperparameters λ, it is possible to replace one of these rules by four new rules, with a total of three attributes and 11 antecedents, necessary to classify correctly a single additional vector, a clear indication that overfitting occurs.

One hundred percent reliability of all rules is achieved after optimization of $\mathcal{R}^{(2)}$ rules with increasing $\gamma \geq 0$ and

minimization of (16). The smallest value of γ for which all rules do not make any errors is found. This set of rules leaves 11 vectors—eight virginica and three versicolor—as unclassified

$$\mathcal{R}_1^{(3)}\text{: setosa if } (X_3 < 2.9) \quad (100\%)$$
$$\mathcal{R}_2^{(3)}\text{: versicolor if } (X_3 \in [2.9, 4.9] \wedge X_4 < 1.7) \quad (100\%)$$
$$\mathcal{R}_3^{(3)}\text{: virginica if } (X_3 \geq 5.3 \vee X_4 \geq 1.9) \quad (100\%).$$

The vectors rejected by $\mathcal{R}^{(3)}$ rules may be classified by rules $\mathcal{R}^{(2)}$, but the reliability of classification for the vectors in the $\mathcal{R}^{(2)} \setminus \mathcal{R}^{(3)}$ border region is rather low: with $p = 8/11$ they should be assigned to the virginica class and with $p = 3/11$ to the versicolor class. For this small dataset, the true probability distributions of leaf sizes for the two classes of the iris flowers certainly overlap, so a more precise answer is impossible.

The Iris example is too simple to see the full advantage of applying optimization and probabilistic evaluation of rules, since the number of parameters to optimize is small and optimal accuracy (98%) is achieved with crisp rules. For cases near the decision border between iris virginica and iris versicolor, more realistic probabilities $p(C|\mathbf{X}; M)$ are calculated using (23). The natural uncertainties here are ± 0.1, equal to the accuracy of measurements. Six vectors near the virginica/versicolor decision border have probabilities between 0.5 and 0.75, the remaining vectors have higher probabilities [11].

D. Comparison

We have used the Iris example for pedagogical reasons only. Reclassification accuracies (in-sample accuracies for the whole dataset) of rules derived by several rule-extraction systems are reported in Table 2. Complexity and reclassification accuracy of rules found by the different methods give some idea about their relative merits. Examples of statistical estimation of accuracy (out-of-sample accuracy) are given for larger datasets with separate test parts later in this paper. The number of rules and conditions does not characterize fully the complexity of the set of rules, since fuzzy rules have additional parameters. The "else" condition is not counted as a separate rule.

The neurofuzzy ReFuNN [13], [123] and NEFCLASS systems [119] belong to the best known of its kind. Linguistic variables used by these systems are based on three equally distributed fuzzy sets for each feature, limiting the accuracy that may be achieved with such rules. The best seven fuzzy rules of NEFCLASS classified correctly 96.7% of data. This system was not able to reduce the number of features automatically, but if used with the last two Iris features, it will give the same performance using only three best rules (out of nine possible) with six conditions. ReFuNN found nine rules with 26 conditions and could also benefit from feature selection. Increasing the number of fuzzy linguistic variables to five per feature leads to 104 rules and 368 conditions. The main purpose of building rule-based systems, i.e., comprehensibility of data description, is lost if the number of rules is large. The poor result of the FuNe-I neurofuzzy system [120] is also due to the use of fuzzy linguistic variables based on fixed partition of the data range.

Table 2
Number of Rules (NR), Number of Conditions (NC), and Number of Features Used (NF) by Rules Extracted for the Iris Dataset by Different Systems. Rules Are Either Crisp (C), Fuzzy (F), Rough (R), or Weighted (W)

Method	NR/NC/NF features	Type	Reclassification accuracy
ReFuNN [13]	9/26/4	F	95.7
ReFuNN [13]	104/368/4	F	95.7
NEFCLASS [119]	7/28/4	F	96.7
NEFCLASS [119]	3/6/2	F	96.7
FuNe-I [120]	7/-/3	F	96.0
Grobian [121]	118/-/4	R	100.0
GA+NN [122]	6/6/4	W	100.0
C-MLP2LN [11]	2/2/1	C	95.7
C-MLP2LN [11]	2/2/2	C	96.0
C-MLP2LN [11]	2/3/2	C	98.0
SSV [11]	2/2/2	C	98.0

Rough sets do not produce a comprehensible description of this simple data, creating a large number of rules. The Grobian rough set system [121] used 118 rules for perfect classification, reaching only 91%–92% in tenfold cross-validation tests. Earlier application of rough sets to the Iris data [124] gave very poor results (77% accuracy), probably because four linguistic attributes per feature were used. This shows again the importance of optimization and the use of context-dependent linguistic variables instead of *ad hoc* partitions of input features. A combination of evolutionary optimization with an MLP network [122] also finds 100% correct rules that overfit the data and will perform poorly in cross-validation tests. Thus, even such simple data may be rather difficult to handle for some rule-extraction systems.

IX. Illustrative Applications

This section provides examples of interesting logical rules discovered using computational intelligence methods that were used to analyze datasets available in standard repositories [118] and collected in various projects. Analysis of each dataset illustrates different aspects of the usefulness of rule-extraction algorithms. Most datasets contain mixed nominal–numerical features, but some contain only symbolic information, or only numerical data. Some datasets are small (around 100 cases), and some are large (tens of thousands of cases). Some data seem to contain only trivial knowledge, other data contain useful knowledge. In some cases crisp logical rules are sufficient, while in a few cases fuzzy rules seem to be more appropriate. A real-life example leading to the construction of an expert system for psychometric applications is provided. An example of relational rules for an approximation problem is also given.

Rule-extraction methods should not be judged only on the basis of the accuracy of the rules, but also on their simplicity and comprehensibility. Unfortunately, different rules are generated on each cross-validation partition, making it difficult to estimate expected statistical accuracy of rules. The simplest sets of rules are usually quite stable; that is, the same rules are extracted in different cross-validation tests,

and reclassification accuracy for such rules is close to cross-validation estimations. The best comparison of expected accuracy is offered on large datasets with separate test parts.

A. Mushrooms

The first dataset contains descriptions of 8124 samples of mushrooms, corresponding to 23 species of gilled mushrooms of the Agaricus and Lepiota family [118]. Descriptions were based on *The Audubon Society Field Guide to North American Mushrooms*. Each mushroom sample is labeled as edible or nonedible, meaning either poisonous or not recommended for eating. The guide states clearly that there is no simple rule for determining the edibility of a mushroom—no rule like "leaflets three, let it be" for poisonous oak and ivy.

Properties of mushrooms include cap shape, surface, and color; gill attachment, spacing, size, and color; stalk shape, root, surface, and color above and below the ring; ring type and number; veil type and color; odor; and spore print color, as well as information about the type of mushroom population and habitat. Altogether 22 symbolic attributes are given for each mushroom sample, and each attribute may take up to 12 different values, which is equivalent to 118 logical features indicating presence or absence of a specific feature. For example, mushroom may have one of the nine odor types: almond, anise, creosote, fishy, foul, musty, pungent, spicy, or none.

In 51.8% of the cases, the mushrooms are edible, the rest being nonedible (mostly poisonous). The problem of dividing them into these two categories is linearly separable; thus, perfect classification may be achieved using linear discrimination or other standard statistical techniques. Unfortunately, such methods do not help to understand why some mushrooms should not be eaten. A single neuron may learn correctly all training samples, but the resulting network has many nonzero weights and is difficult to analyze. Using the C-MLP2LN algorithm with the cost function (13), the following disjunctive rules for nonedible mushrooms have been discovered [11]:

$$\mathcal{R}_1 : \text{odor} = \neg(\text{almond} \vee \text{anise} \vee \text{none})$$

$$\mathcal{R}_2 : \text{spore} - \text{print} - \text{color} = \text{green}$$

$$\mathcal{R}_3 : \text{odor} = \text{none} \wedge \text{stalk} - \text{surface} - \text{below} - \text{ring} = \text{scaly} \wedge (\text{stalk} - \text{color} - \text{above} - \text{ring} = \neg\text{brown})$$

$$\mathcal{R}_4 : \text{habitat} = \text{leaves} \wedge \text{cap} - \text{color} = \text{white}.$$

Rule $\mathcal{R}_1$ misses 120 nonedible cases (98.52% accuracy), and says that edible mushrooms from the Agaricus and Lepiota family have no odor, or it must be of the almond or anise type, otherwise they are nonedible. Adding rule $\mathcal{R}_2$ leaves 48 errors (99.41% accuracy), adding the third rule leaves only eight errors (99.90% accuracy), and all four rules $\mathcal{R}_1$ to $\mathcal{R}_4$ distinguish between edible and nonedible mushrooms correctly. For large values of the weight-decay regularization parameter λ_1 in (13), only one rule with a single attribute (odor) is produced, while for smaller values of λ_1, a second attribute (spore-print-color) contributes to the activation of the first neuron. 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 eight cases.

Table 3
Summary of Rule-Extraction Results for the Mushroom Dataset, Giving Number of Rules (NR), Conditions in All Rules (NC), Number of Features Used (NF), and Reclassification Accuracy of the Rules

Method, reference	NR/NC/NF	accuracy %
RULENEG [85]	300/8087/–	91.0
REAL [83]	155/6603/–	98.0
C-MLP2LN, SSV [11]	1/3/1	98.5
RULEX [97]	1/3/1	98.5
C-MLP2LN, SSV [11]	2/4/2	99.4
Successive Regulariz. [125]	1/4/2	99.4
DEDEC [90]	26/26/–	99.8
C4.5 (decision tree)	3/3/5	99.8
Successive Regulariz. [125]	2/22/4	99.9
C-MLP2LN, SSV [11]	3/7/4	99.9
Successive Regulariz. [125]	3/24/6	100.0
TREX [4]	3/13/–	100.0
C-MLP2LN [11]	4/9/6	100.0
SSV [11]	4/9/5	100.0

For the mushroom dataset, the SSV tree has found a 100% accurate solution that can be described as four logical rules using only five attributes. The first two rules are identical to the rules given above, but the remaining two rules are different, using "gill-size" instead of stalk and cap-related attributes. Since the last two rules cover only a small percentage of all cases, many equivalent descriptions are possible. SSV rules give perhaps the simplest set of rules found so far

$$\mathcal{R}_1 : \text{odor} = \neg(\text{almond} \vee \text{anise} \vee \text{none})$$

$$\mathcal{R}_2 : \text{spore} - \text{print} - \text{color} = \text{green}$$

$$\mathcal{R}_3 : \text{gill} - \text{size} = \text{narrow} \wedge (\text{stalk} - \text{surface} - \text{above} - \text{ring} = (\text{silky} \wedge \text{scaly}) \vee \text{population} = \text{clustered}.$$

This is the simplest systematic logical description of the mushroom dataset and, therefore, may be used as a benchmark for other rule-extraction methods. Odor is the most important attribute, and perhaps for that reason animals need such a good sense of smell. If odor is removed from the description of mushroom samples, 13 rules are needed to reach 100% correct classification.

The same logical rules have also been derived using as few as 10% of all data for training [125]; therefore, results from cross validation should be identical to the results given in Table 3. Some authors derive conjunctive rules for edible mushrooms (negation of the disjunctive rules for nonedible mushrooms presented above), reporting lower number of rules with the same accuracy; therefore, the total number of conditions is a more reliable measure of the complexity of rule sets. This example illustrates how important the simplicity of the rules is. Although statistical or neural methods may classify this data perfectly, logical rules derived here give probably the most comprehensible description of the data.

B. The Ljubljana Cancer Data

The second example involves prediction of the breast cancer recurrence. The data have been collected at University Medical Centre, Institute of Oncology, Ljubljana (formerly Yugoslavia) [118], and contains 201 cases of patients that

Table 4
Ljubljana Cancer Dataset, Tenfold Cross Validation and Reclassification Accuracy Results, in Percentages

Method/Reference	10-fold CV
C-MLP2LN/SSV, 1 rule [11]	76.2±0.0
SSV, 3 rules [11]	73.5±0.9
MLP + BP [131]	73.5±9.4
FSM, 20 fuzzy rules	72.0±8.3
CART [131]	71.4±5.0
Default	70.3
LERS (rough rules) [130]	69.4
Naive Bayes rule [131]	69.3±10.0
AQ15 [127]	66-72
Weighted network [128]	68-73.5
	Reclassification
Assistant-86 [129]	78.0
C-MLP2LN, 2 rules [11]	78.0
SSV, 3 rules [11]	77.6
SSV, 1 rule [11]	76.2

Table 5
Results From the Tenfold Cross Validation and Reclassification for the Wisconsin Breast Cancer Dataset

Method/Reference	Accuracy %
k-NN, k=3, Manhattan [11]	97.0±2.1
Fisher LDA [134]	96.8
MLP+backprop [134]	96.7
LVQ [134]	96.6
Naive Bayes [134]	96.4
DB-CART [135]	96.2
CART (dec. tree) [135]	93.5
Quadratic DA [134]	34.5
FSM, 12 fuzzy rules [11]	96.9±1.4
SSV, 3 crisp rules [11]	96.3±0.2

Results from reclassification	Accuracy %	Rules/type
C-MLP2LN [11]	99.0	5, C
C-MLP2LN [11]	97.7	4, C
SSV [11]	97.4	3, C
NEFCLASS [119]	96.5	4, F
C-MLP2LN [11]	94.9	2, C
NEFCLASS [119]	92.7	3, F

did not suffer from recurrence of cancer (70.3%) and 85 that had recurrence (29.7%). There are nine attributes, including age (nine bins), tumor size (12 bins), number of nodes involved (13 bins, starting from 0–2), degree of malignancy (1, 2, or 3), and information about menopause, which breast and breast quadrant were affected, and whether radiation treatment has been applied.

These data have been analyzed using a number of algorithms (e.g., [126]–[131]). The results are rather poor, as should be expected, since the data attributes are not very informative. Several ML methods gave predictions that are worse than those of the majority classifier (70.3%), as shown in Table 4. In general, for small datasets, the more complex the model, the worse the results in cross-validation tests.

Strong regularization of the C-MLP2LN, or strong pruning of the SSV decision tree, generates most often a single logical rule for the recurrence events.

```
IF involved nodes ∉ [0,2] ∧ Degree − malignant = 3
  THEN recurrence ELSE no-recurrence.
```

This rule contains rather trivial knowledge (at least trivial for the medical doctors): recurrence of the breast cancer is expected if the number of involved nodes is bigger than minimal (not in [0,2] bin) and the cancer is highly malignant (degree $= 3$). Overall reclassification accuracy of this rule is 76.2% and, since it is stable, reclassification results on the whole data are identical to average cross-validation values. Unfortunately, this rule covers only 37 cases, correctly recognizing 27 out of 85 recurrence cases (sensitivity $= 31.8\%$), and covering incorrectly ten cases from the no-recurrence class (precision of the rule is 73%). Perhaps after longer observation time, recurrence of cancer may also be expected in these cases. The else condition has 95.0% specificity and 76.7% precision.

Such a simple rule cannot overfit the data, and may be found in many (but not in all) cross-validation partitions. A more complex set of three rules obtained using SSV achieves 77.6% reclassification accuracy, but in the tenfold cross-validation tests, the average is only 73.5% (worst result -0.8%, best $+1.0\%$), i.e., only a few percent above the default value, indicating that these rules are already too complex and overfit the data.

A CART decision tree gives 77.1% in the leave-one-out tests [126], but the tenfold cross-validation result quoted in Table 4 is much lower, since the rules are unstable. For small and noisy data, Breiman advocates generating many bootstrap subsets of the data and averaging over all models created on these subsets [50]. For the Ljubljana cancer data, averaging over ten trees created on different subsets of the data improves the results by about 3% at the expense of losing interpretability. Instead of averaging over decision trees or rule-based models, the best approach to find good rules is to select the most frequent (and thus most stable) rule. Another solution is to generate many alternative sets of rules with similar accuracy (for example, a forest of decision trees [43]) and ask the domain expert to select interesting rules.

It would be difficult to improve upon the results of these simple rules, which are easily understood by anyone. We doubt that there is any more information in this dataset. Most methods give significantly lower accuracy using more complex models. For example, an FSM with 20 fuzzy rules (average for all cross-validation) gives results that are only slightly better than the default accuracy. LERS [130], an ML technique based on rough sets, gave after optimization almost 100 "certain" rules and about the same number of "possible" rules, achieving accuracy that is below the majority rate. Although it certainly is not the limit of accuracy for rough set systems, it shows that rule-based systems may also easily overfit the data. A large number of rules will usually lead to poor generalization, and the insight into the knowledge hidden in the data will be lost.

C. Wisconsin Breast Cancer Data

The Wisconsin breast cancer dataset [132] is one of the favorite benchmark datasets for testing classifiers (Table 5). Properties of cancer cells were collected for 699 cases, with

458 benign (65.5%) and 241 (34.5%) malignant cases of cancer. They include estimation of the uniformity of cell size and shape, epithelial cell size, and six other attributes, all taking integer values ranging from one to ten. The problem is to distinguish between malignant and benign cancer cells.

The simplest rule that has been found using SSV and C-MLP2LN algorithms says the following.

```
IF Uniformity of Cell Size=1 or 2
THEN benign,
ELSE malignant.
```

This rule assigns 41 benign cases to malignant and 12 malignant cases to benign, achieving 92.4% accuracy (sensitivity 91.0%, specificity 95.0%, and precision 97.2%). Two more accurate rules for the malignant class were found using the C-MLP2LN algorithm [11].

```
IF Uniformity of Cell Size ≥ 7∨ Bland
Chromatin ≥ 6
THEN malignant, ELSE benign
```

These rules cover 215 malignant cases and ten benign cases (precision 95.6%), achieving overall accuracy (including the else condition) of 94.9%. Many other sets of rules of slightly higher complexity may be discovered using the forest of decision trees [43].

Using L-units in the C-MLP2LN network, four quite accurate rules for the malignant class were created (their precision is in parenthesis)

$$\begin{aligned}
&\mathcal{R}_1\colon f_4<3\wedge f_5<4\wedge f_7<6\wedge f_9=1 \quad (99.5\%)\\
&\mathcal{R}_2\colon f_2<7\wedge f_5<4\wedge f_7<6\wedge f_9=1 \quad (99.8\%)\\
&\mathcal{R}_3\colon f_2<7\wedge f_4<3\wedge f_7<6\wedge f_9=1 \quad (99.4\%)\\
&\mathcal{R}_4\colon f_2<7\wedge f_4<3\wedge f_5<4\wedge f_7<6 \quad (99.4\%).
\end{aligned}$$

For simplicity, the names of the features are replaced here with their numbers in the description of the data (f_1 is clump thickness; see [118] for full description). Including the else condition, these rules classify all data with 97.7% accuracy. Only five malignant cases were misclassified as benign (FNs) and 11 benign as malignant (FP). A fuzzified version of these rules [using (23) corresponding to the Gaussian dispersion of inputs equal ± 0.5] predicts with almost 100% confidence that these cases belong to the wrong class. This is an indication that the dataset may contain a few misclassified cases.

Many other sets of rules have been discovered for this dataset [11], [133]. An optimized set of five rules [using (16)] for the malignant cancer cells gives

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

These rules classify only one benign vector as malignant ($\mathcal{R}_1$ and $\mathcal{R}_5$, the same vector), and the else condition for the benign class makes six errors, giving 99.0% overall accuracy.

Fuzzy rules do not lead to higher accuracy, nor do they offer a better insight into the structure of this data. The NEFCLASS neurofuzzy system [119] with three trapezoidal membership functions per input feature generated four rules, and the "best per class" rule learning gave only 80.4% correct answers. If only two membership functions per feature are used, a much better reclassification accuracy of 96.5% is obtained using the same number of fuzzy rules. The FSM neurofuzzy system generated 12 rules with Gaussian membership functions, providing 97.8% on the training and 96.5% on the test part in tenfold cross-validation tests. Crisp rules are more accurate and understandable in this case.

Creating a forest of heterogeneous decision trees [43], a single rule with 97.3% accuracy (sensitivity $= 97.9\%$ and specificity $= 96.9\%$) has been found. This rule identified one of the training vectors as a good prototype for the malignant case

$$\begin{aligned}
&\text{IF } d(\mathbf{X},\mathbf{V}_{303})<62.7239 \text{ THEN malignant}\\
&\text{ELSE} \qquad\qquad \text{benign}
\end{aligned}$$

where $d(\mathbf{X},\mathbf{V}_{303})$ is the Euclidean distance from this vector. Prototype-based rules are little known so far and computationally more costly to find, but certainly for some data, they may be simple and accurate.

D. Diabetes

The "Pima Indian Diabetes" dataset [118] is also frequently used as benchmark data [89], [134]–[136]. All patients were females, at least 21 years old, of Pima Indian heritage. Seven hundred sixty-eight cases have been collected, 500 (65.1%) healthy and 268 (34.9%) with diabetes. Eight attributes describe age, number of times pregnant, body mass index, plasma glucose concentration, diastolic blood pressure, diabetes pedigree function, and other medical tests.

This dataset was used in the Statlog project [89], with the best tenfold cross-validation accuracy around 77.7% obtained by logistic discriminant analysis. Our estimation of variance on cross-validation partitions is about $\pm 5\%$. Histograms overlap strongly and do not provide a useful starting point for the definition of linguistic variables for this dataset, but neural L-units and SSV separability criterion (8) provide good cutoff points. Simplest rules use only two attributes, the "plasma glucose concentration" (PGC) and the body mass index (BMI), weight in kg/(height in m)^2. A single rule obtained from the C-MLP2LN network and SSV tree

```
IF PGC > 144.5 THEN diabetes
ELSE healthy
```

has an accuracy rate of 74.9% and, since the rule is quite stable, derived as the best for all partitions of data, reclassification and cross-validation results are identical. Unfortunately, this rule recognizes only 122 out of 268 cases with di-

Table 6
Results From the Tenfold Cross Validation and Reclassification for the Pima Indian Diabetes Dataset; Accuracy in Percentages

Method/Reference	Accuracy
Logistic discrimination [89]	77.7
DIPOL92 [89]	77.6
LDA [89]	77.5
IncNet neural net [137]	77.2±3.3
FDA [89]	76.5
MLP [89]	76.4
kNN, k=20, Euclidean	75.9±3.4
LVQ [89]	75.8
RBF neural net[89]	75.7
SSV, 3 rules	75.3±4.8
Naive Bayes [89]	75.3
DB-CART, 33 nodes [135]	74.4
FSM, 50 Gaussian [11]	73.7±3.1
CART, 11 nodes [135]	73.7
C4.5 [89]	73.0
CART [89]	72.8
Kohonen SOM [89]	72.2
kNN, k=1 [89]	71.9
Reclasssification	
C-MLP2LN, 2 rules [11]	77.7
C-MLP2LN, 1 rule [11]	75.0

abetes correctly (sensitivity $= 45.5\%$, specificity $= 90.6\%$). Two rules found by the SSV decision tree

$$\texttt{IF } (\mathrm{PGC} > 144.5) \vee (\mathrm{PGC} > 123.5 \wedge \mathrm{BMI} > 32.55)$$
$$\texttt{THEN diabetes, ELSE healthy}$$

achieve overall accuracy of 76.2%, with sensitivity $= 60.8\%$ and specificity $= 84.4\%$. On some cross-validation partitions, slightly different rules are found with the same pruning or regularization parameters; therefore, the tenfold cross-validation accuracy of these rules is $75.3 \pm 4.8\%$. It is difficult to improve significantly upon these results using fuzzy rules. For example, using an FSM neurofuzzy system with about 50 Gaussian functions optimized for each tenfold cross-validation partition gives 86.4% accuracy on the training part, but only $73.7 \pm 3.1\%$ on the test part. An incremental network (IncNet [137]) using six to eight Gaussian functions achieves $77.2 \pm 3.3\%$ on the test part. Only slightly better results are achieved using linear discrimination. Comparison of the results given in Table 6 is not complete, since the standard deviation of most methods is rather high, and differences of a few percent are not statistically signifcant. We have recalculated some of the results using the Ghostminer software package [138] to provide better comparison.

Statistical discrimination methods work quite well in this case, indicating that a single hyperplane divides the classes in an almost optimal way. Simple logical rules are also quite competitive in this case, allowing for understanding of important factors that determine the diagnosis.

E. The Hypothyroid Data

This is a somewhat larger medical dataset [118], containing screening tests for thyroid problems. The training data have 3772 medical records collected in the first year, and the test

Table 7
Results for the Hypothyroid Dataset

Method	% train	% test
C-MLP2LN [11]	99.89	99.36
CART [139]	99.79	99.36
PVM [139]	99.79	99.33
SSV, 3 rules [11]	99.79	99.33
FSM 10 rules [11]	99.60	98.90
Cascade correlation [140]	100.00	98.5
Best MLP+backprop [140]	99.60	98.5
3-NN, 3 features used [11]	98.7	97.9
Naive Bayes [139]	97.0	96.1
k-NN [139]	100.00	95.3

data have 3428 cases collected in the next year of the study. Most people are healthy (92.47% in the training and 92.71% in the test data), some suffer from primary hypothyroid conditions (about 5%), and some from compensated hypothyroid conditions (about 2.5%). Each record stores information about person's age, a number of binary indicators such as sex, goiter, whether the person is on some medications, is pregnant, had thyroid surgery, tumors, and the level of five hormones, TSH, T3, TT4, T4U, and FTI. Altogether 21 attributes are provided, 15 binary and 6 continuous.

Four simple rules were found by the C-MLP2LN algorithm [11], with 99.68% accuracy on the training set and 99.07% accuracy on the test set. For the primary hypothyroid class, two rules are sufficient (all values of continuous features are multiplied here by 1000)

$$\mathcal{R}_{11} : \mathrm{FTI} < 63 \wedge \mathrm{TSH} \geq 29$$
$$\mathcal{R}_{12} : \mathrm{FTI} < 63 \wedge \mathrm{TSH} \in [6.1, 29) \wedge \mathrm{T3} < 20.$$

For the compensated hypothyroid class, one rule is created

$$\mathcal{R}_2\text{: } \mathrm{FTI} \in [63\,180] \wedge \mathrm{TSH} \geq 6.1 \wedge \text{on thyroxine} = \text{no} \wedge \text{surgery} = \text{no}$$

and the third class (healthy) is covered by the else condition. These rules give 99.68% accuracy on the training set and 99.07% error on the test set. Optimization of these rules leads to a more accurate set (precision is given in parentheses, with the same rule order as above)

$$\mathcal{R}_{11}\text{: } \mathrm{FTI} < 64.27 \wedge \mathrm{TSH} \geq 30.48 \quad (97.06\%)$$
$$\mathcal{R}_{12}\text{: } \mathrm{FTI} < 64.27 \wedge \mathrm{TSH} \in [6.02, 29.53] \wedge \mathrm{T3} < 23.22 \quad (100\%)$$
$$\mathcal{R}_2\text{: } \mathrm{FTI} \in [64.27, 186.71] \wedge \mathrm{TSH} \geq 6.02 \wedge \mathrm{TT4} \in [50\,150.5) \wedge \text{on thyroxine} = \text{no} \wedge \text{surgery} = \text{no} \quad (98.96\%).$$

The else condition for healthy cases has 100% precision on the training set. These rules make only four errors on the training set (99.89%) and 22 errors on the test set (99.36%). The rules are conservative, assigning healthy persons (all errors are very close to the decision border) to one of the hypothyroid problem groups. Rules of similar quality have been found by Weiss and Kapouleas [139] using a heuristic version of the predictive value maximization (PVM) method and using the CART decision tree. The differences among PVM, CART, and C-MLP2LN for this dataset are rather

small (Table 7), but other methods, such as well-optimized MLP (including evolutionary optimization of network architecture [140]) or cascade correlation neural classifiers, give results that are significantly worse. The poor results of k-NN are especially worth noting, showing that in this case, despite a large number of reference vectors, similarity-based methods are not competitive. Ten fuzzy rules obtained using FSM with Gaussian membership functions are also less accurate than the three crisp rules presented above.

The C-MLP2LN solution seems to be close to optimal [141]. Similar rules were found with the SSV decision tree (again, $\mathcal{R}_1$ is for the primary hypothyroid class, $\mathcal{R}_2$ for the compensated hypothyroid)

$$\mathcal{R}_1\text{: FTI} < 64.72 \wedge \text{TSH} > 6.05 \wedge \text{thyroid} - \text{surgery} = \text{no}$$

$$\mathcal{R}_2\text{: FTI} > 64.72 \wedge \text{TSH} > 6.05 \wedge \text{TT4} < 150.5 \wedge \text{thyroid} - \text{surgery} = \text{no} \wedge \text{on} - \text{thyroxine} = \text{no}$$

ELSE healthy.

The test set accuracy of these rules is 99.33%. A heterogeneous version of the SSV tree [43] found modified rule $\mathcal{R}_1$ (primary hypothyroid class)

$$\mathcal{R}_1\text{: FTI} < 64.72 \wedge \text{TSH} > 6.05 \wedge (\text{T3} < 11.5 \vee d(\mathbf{X}, \mathbf{V}_{2917}) < 1.105\,31)$$

involving Euclidean distance from vector number 2917. This modified set of rules makes only six errors on the training set (99.84%) and 21 errors on the test set (99.39% accuracy). These rules have been found with a fully automatic rule-extraction approach.

The results are summarized in Table 7. It is worth noting that the error of the best neural network classifiers is still twice as large (1.5%) as the error made by these simple rules. Forcing sharp division into three output classes creates the need for sharp decision borders that fuzzy and neural systems cannot easily provide. This may be the reason for excellent results obtained by crisp logic sets of rules for medical data. Thus, extracted rules can expose logical structure hidden in the data.

F. Hepatobiliary Disorders

The next example shows that crisp logic rules are not always successful but may be helpful to discover problems with the data. The hepatobiliary disorders dataset contains medical records of 536 patients admitted to a university-affiliated Tokyo-based hospital, with four types of hepatobiliary disorders: alcoholic liver damage, primary hepatoma, liver cirrhosis, and cholelithiasis. The records include results of nine biochemical tests and the sex of the patient.

These data have been used in several publications [116], [142]–[144]. They have been divided into 373 training cases (with class distribution of 83, 127, 89, and 74) and 163 test cases (with class distribution of 33, 51, 35, and 44) [142]. Three fuzzy sets per each input were assigned according to the recommendation of the medical experts. A fuzzy neural network was constructed and trained until 100% correct answers were obtained on the training set. The accuracy on the test set varied from less than 60% to a peak of 75.5%. Although we quote this result in Table 8, from the methodological point of view they are not correct, since the best network has been selected by looking at the test set results. Fuzzy rules equivalent to the fuzzy network were derived, but their accuracy on the test set was not given. Mitra *et al.* [143] used a knowledge-based fuzzy MLP system, achieving test set accuracy between 33% and 66.3%, depending on the actual fuzzy model used.

Simple rules created by decision trees or C-MLP2LN procedure have quite low accuracy, reaching only 50%–60%. The C-MLP2LN algorithm found 49 crisp logic rules resulted in 83.5% accuracy on the training and 63.2% on the test set [11]. Optimizing the rules did not improve these results significantly. The best results were obtained from decision trees without any pruning, leading to about 100 rules and 68%–75% accuracy on the test set. The k-nearest-neighbors algorithm achieves best results for $k = 1$, also indicating that decision borders are very complex and many logical rules will be required.

Fuzzy rules derived using the FSM network, with Gaussian as well as with triangular functions, gave an average accuracy of 75.6%–75.8%. A neurofuzzy FSM network used over 100 neurons to achieve this. Rotating these functions (i.e., introducing a linear combination of inputs to the rules) did not improve this accuracy. An attempt to find good prototypes was made in [18], optimizing the selection of a distance function (Canberra distance was selected, summing $|X_i - Y_i|/|X_i + Y_i|$ contributions from all features), and using feature selection (four features were removed). The algorithm generated 57 prototypes (out of 373 training vectors), but the prototype-based rules achieved only 64.2% accuracy on the test set. Tenfold cross-validation tests on the mixed data (training plus test data) gave similar results.

Many methods give rather poor results on this dataset, including variants of instance-based learning (IB2–IB4, except for the IB1c, which is designed specifically to work with continuous input data), statistical methods (Bayes, LDA), and pattern recognition methods (LVQ). The best results were obtained with the K* method based on algorithmic complexity optimization, giving 78.5% on the test set, and the k-nearest-neighbors with Manhattan or Canberra distance function, $k = 1$, and selection of features, giving 83.4% accuracy (for details, see [18], [145]). Various results for this dataset are summarized in Table 8.

For these data, rules in any form do not seem to work well, indicating that classes overlap strongly. The best one can do is to identify the cases that can be classified reliably, and assign the remaining cases to pairs of classes [146]. The data seem to be very noisy, but results are well above the default (31% for the majority class in the test data). One may expect such behavior of the classification methods for data with a high degree of randomness, containing pairs of identical or very similar vectors. For example, results of tests made on the same patient taken at different times will produce very similar vectors. Indeed, visual inspection of the data using multidimensional scaling would show many paired vectors, including 26 identical pairs of vectors in the training data.

Table 8
Results for the Hepatobiliary Disorders (Taken From [11]). Accuracy in Percentages on the Training and Test Sets

Method	Training set	Test set
1-NN, Canberra, f. selec	77.2	83.4
1-NN, weighted (ASA)	83.4	82.8
K* method	–	78.5
kNN, k=1, Manhattan	79.1	77.9
FSM, Gaussian functions	93	75.6
FSM, 60 triangular functions	93	75.8
IB1c (instance-based)	–	76.7
FSM, Gaussian functions	93	75.6
C4.5 decision tree	94.4	75.5
Fuzzy neural network	100	75.5
Cascade Correlation	–	71.0
MLP with RPROP	–	68.0
Best fuzzy MLP model	75.5	66.3
C4.5 decision rules	64.5	66.3
LDA (statistical)	68.4	65.0
FOIL (inductive logic)	99	60.1
1R (rules)	58.4	50.3
Naive Bayes	–	46.6
IB2-IB4	81.2-85.5	43.6-44.6

Table 9
Summary of Results for the NASA Shuttle Dataset

Method/Reference	Train	Test
SSV, 32 rules [11]	100.00	99.99
NewID dec. tree	100.00	99.99
FSM, 17 rules [11]	99.98	99.97
k-NN + feature sel. [11]	–	99.95
C4.5 dec. tree [89]	99.96	99.90
k-NN [89]	–	99.56
RBF [89]	98.40	98.60
MLP+BP [89]	95.50	96.57
Logistic discrimination [89]	96.07	96.17
Linear discrimination [89]	95.02	95.17

G. NASA Shuttle

The shuttle dataset from NASA contains nine continuous numerical attributes related to the positions of radiators in the space shuttle [118]. There are 43 500 training vectors and 14 500 test vectors, divided into seven classes in a very uneven way: in the training set, about 80% of vectors belong to class 1, and only six examples are from class 6. These data have been used in the Statlog project [89]; therefore, the results obtained with many classification systems are available for comparison (Table 9).

An MLP network trained with backpropagation algorithm reached an accuracy of 95.5% on the training set and 96.57% on the test set. An RBF network works better here, reaching 98.60% on the test set. k-nearest-neighbor is very slow in this case, requiring all 43 500 training vectors as reference for computing distances, reaching 99.56% on the test set, but with feature selection improving to 99.95%.

An FSM network with rectangular membership functions and an SSV decision tree have been used with great success for these data [11]. Initializing the FSM network gives seven nodes (one per class), already achieving 88% accuracy. An optimized set of 17 crisp logic rules generated by an FSM network makes only three errors on the training set (99.99% correct), leaving eight vectors unclassified, and no errors on the test set, leaving nine vectors unclassified (99.94% correct). For example, the rules for the third class are

$$F2 \in [-188.43, -27.50] \wedge F9 \in [1, 74]$$
$$F2 \in [-129.49, -21.11] \wedge F9 \in [17, 76].$$

After small Gaussian fuzzification of inputs (as described in Section VII), only three errors and five unclassified vectors are left for the training, and one error is made (with the probability of correct class for this case being close to 50%) and three vectors left unclassified for the test set.

The 17 rules created by SSV decision tree gave even better results. They are 100% correct on the training data and make only one error on the test set (accuracy equal to 99.99%). A similar accuracy has been achieved by the NewID decison tree (descendant of the ID3 tree, [89]), although in the Statlog project, this tree has not been among the best three methods for any other of the 22 datasets analyzed. Results of the C4.5 decision tree are significantly worse.

Logical rules provide most accurate and quite simple (two to three rules per class) description of the NASA shuttle dataset. Technical applications may quickly produce very large number of data samples, but rule-extraction systems, especially those based on decision trees, handle such data efficiently.

H. Psychometry

The Minnesota multiphasic personality inventory (MMPI) test is used frequently to evaluate psychological characteristics reflecting social and personal maladjustment, including psychological dysfunction. The test consists of 550 questions, with three possible answers (yes, no, do not know) to each question. Hundreds of books and papers have been written on the interpretation of this test (see [147]). Many computerized versions of the MMPI test exist to assist in information acquisition, but evaluating the results is still done by an experienced clinical psychologist. Since the number of well-trained psychologists is limited, providing an automatic support for psychological diagnosis is important. In fact, this project was initiated by psychologists looking for computational tools that could speed up the process of initial evaluation of questionnaires and support their diagnostic decisions.

Linear combinations of the raw MMPI test answers are used to compute 14 real-valued coefficients, called psychometric scales. These coefficients are often displayed as a histogram (called a psychogram), allowing skilled psychologists to diagnose specific problems, such as neurosis, drug addiction, or criminal tendencies. The first four coefficients are used as control scales, measuring the willingness and the consistency of answers, which determine the validity of the analyzed test. The remaining coefficients are used for the so-called clinical scales. These scales were developed to measure tendencies toward hypochondria, depression, hysteria, psychopathy, paranoia, schizophrenia, etc. A large number of simplification schemes have 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 three Goldberg coefficients [147].

A rule-based system is most desirable because a detailed interpretation, including a description of personality type, may be associated with each diagnosis, depending on the rule conditions. Two datasets were used in this study [113], [148], one for women, with 1027 cases belonging to 27 classes (normal, neurotic, drug addicts, schizophrenics, psychopaths, organic problems, malingerers, etc.) determined by expert psychologists, and the second for men, with 1167 cases and 28 classes. The psychometric data came from the Academic Psychological Clinic of Nicolaus Copernicus University, Poland. Answers to the questionnaires have been combined into 14 psychometric coefficients (scales). This reduced data was analyzed by the C4.5 classification tree [42] and the FSM neurofuzzy network to generate an initial set of logical rules.

For the dataset containing tests for women, the C4.5 rules algorithm created 55 rules, achieving 93.0% correct responses. Since it is impossible to estimate the accuracy of the psychometric coefficients used as input values, Gaussian uncertainties were introduced [as described in Section VII, (22)]. Their optimal values (minimizing the number of errors on the training set) were between 1% and 1.5% of the feature range. This procedure allows for an evaluation of the probabilities and improves C4.5 rules reclassification results to 93.7%. An FSM network with rectangular membership functions generated 69 rules, agreeing in 95.4% of the cases with the diagnoses by human experts. Gaussian fuzzification increases the accuracy of these rules to 97.6%. Larger input uncertainties, up to 5% of the feature range, lead to roughly the same number of classification errors as the original crisp rules, but provide softer evaluation of possible diagnoses, assigning nonzero probabilities to classes that were not covered by the slightly fuzzified rules, and reminding the experts of the possibility of alternative diagnoses.

For the dataset containing tests for men, the C4.5 rules algorithm created 61 rules that gave 92.5% correct answers (93.1% after fuzzification), while FSM generated 98 rules, giving 95.9% accuracy, and after fuzzification 96.9%. Statistical estimation of generalization by tenfold cross validation gave 82%–85% correct answers with FSM (crisp unoptimized rules) and 79%–84% correct answers with C4.5. Fuzzification improves FSM cross-validation results to 90%–92%. Results from the IncNet, a neural network model [137], are significantly more accurate, reaching 93%–95% in cross-validation tests, compared with 99.2% for classification of the whole dataset using the same model [113].

There are two to three rules per class, each involving two to nine attributes. For most classes, there were only a few errors, and it is quite probable that they were due to the psychologists' interpretation of the psychogram data. Two classes, organic problems and schizophrenia, were especially difficult to diagnose, since their symptoms are easily confused with each other. Schizophrenia patients may have symptoms similar to that of neurosis, paranoia, drug addiction, and a few other classes. Psychologists have assigned a detailed interpretation to each of these rules. Fuzzification leads to additional adjectives in verbal interpretation, such as "strong tendencies" or "typical." An expert system using these rules should be evaluated by clinical psychologist in the near future. A typical rule has the form

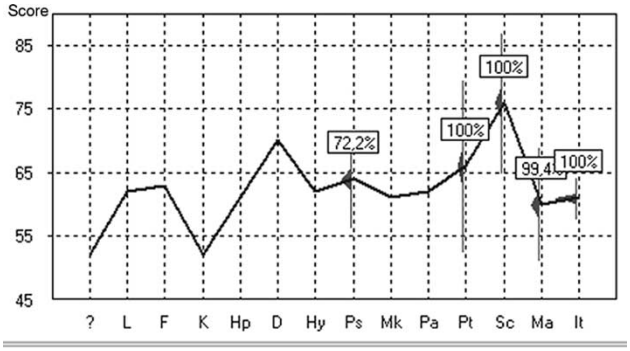

Fig. 10. Psychogram with rule conditions and fuzzified inputs (psychometric scales) displayed. Control scales start with "?," and the clinical scales with Hp (hypochondria). The match between rule conditions and actual values calculated from the MMPI test is shown in percentages.

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

where f_7 is the hysteria scale, f_{12} is the schizophrenia scale, and f_{14} is the social introversion scale value. An example of a psychogram with rule conditions shown as vertical bars is shown in Fig. 10. The rule has five conditions and the actual case is accepted by that rule with 71.8% probability, calculated with an assumption of Gaussian uncertainties shown on the vertical bars for each condition. The rule condition for the psychostenia (Ps) scale fits with only 72.2% to the measured value, showing that this value is close to the boundary of linguistic variable interval.

Although logical rules are not the most accurate on these data, they certainly are most useful. It is not clear how accurate the psychometric evaluation are and how to measure the errors in such applications. The class labels for the cases in the database have been provided by a single clinical psychologist. If several independent diagnoses were made, the psychometric decision support system could use a voting committee approach to reach more reliable conclusions, or could be trained with probabilities of the diagnoses.

I. Automobile Gas Mileage Prediction

Each of the previous examples concerned classification problems, where the predicted class memberships were discrete and crisp propositional logic rules worked rather well. Rule-based description may also be used for approximation problems, when variables that should be predicted are continuous. Crips rule conditions with spline-type functions used as conditions may be used for local approximation of complex functions. In the automobile gas mileage prediction example described below, fuzzy relational rules are extracted.

A common feature of the vast majority of data-driven rule-based fuzzy models presented in the literature is the partitioning of the input space, based on the assumption of the independence between the data components. In many cases, this assumption does not hold, because complex

systems, encountered in real environments, contain input attributes that exhibit some degree of mutual dependency. An implementation of a data-driven fuzzy model incorporating dependencies between its input variables into the process of approximate reasoning, and based on the idea of relational rule partitioning [149], was proposed in [150]. In the context of data understanding, exposing the dependencies offers more insight into the data structure, thereby providing more realistic representation of many complex systems [149].

Relational fuzzy rules proposed in [150] and [151] employ the following linguistic proposition in order to express the functional relationship of local, approximately linear, mutual dependence between two variables:

$$A_1(X_1) \text{ is positively or negatively correlated with } A_2(X_2) \tag{25}$$

where A_1, A_2 are 1-D fuzzy sets on the universes of X_1 and X_2, respectively. An example of such a proposition is "engine r/min around 2500 is positively correlated with vehicle speed around 70 mi/h."

In this proposition, "around 2500" and "around 70" are fuzzy sets, and "engine r/min" and "vehicle speed" are linguistic variables. For a given data instance, to have a high similarity degree with such a defined linear prototype, it must not only preserve the correlation pattern, but also be sufficiently close to (2500; 70). For example, a data sample (1500; 50) has a smaller degree of similarity than (2000; 60). Although both exhibit the same similarity to the correlation pattern, the latter is more compatible with the notions of "r/min around 2500" and "speed around 70 mi/h." The proposition (25) can be represented in fuzzy set theory by a two-dimensional (2-D) fuzzy set.

An n-dimensional fuzzy set (relation), where $n > 2$, can be decomposed into $n(n-1)/2$ 2-D sets (binary relations). For large n's, the inclusion of all possible propositions (25) in a single rule would render the proposed approach useless. However, the approximately linear character of the discussed relationships requires that only $n-1$ binary relations be used, in order to adequately represent the considered n-dimensional fuzzy set. For more detailed explanation of the theoretical aspects involved in the construction of relational fuzzy rules, see [151].

The process of generating relational fuzzy rule-based data explanations consists of the following two steps.

1) *Initial rule generation.*
   Fuzzy c-means clustering [152] is first performed to find similar data groups (clusters). The locations and shapes of the initial membership functions are estimated using subsets of data whose membership to the corresponding clusters is greater than some threshold membership value. The shape parameters include spreads as well as rotation angles. The shape parameters of the membership functions allow for assessment of strength of the potential linear relationships between the pairs of input variables. The estimated membership functions are employed in the creation of initial relational linguistic rules. The pairing of the input variables in the rule antecedent is performed using the available *a priori* knowledge. When no such knowledge exists, the variables believed to be strongly correlated with each other are paired.
2) *Parameter tuning.*
   The parametric representation of the membership functions, estimated from data, facilitates numerical minimization of the approximation error. This minimization can be performed using any first- or second-order gradient-based optimization method. After this minimization is completed, the refined membership functions are translated into final relational linguistic rules.

This simple example illustrates the idea of using relational fuzzy rules in the analysis of the relationship between automobile fuel consumption and several of its characteristics. The automobile gas mileage prediction is a nonlinear regression problem of modeling fuel consumption based on several vehicle characteristics. The following six numerical attributes specify the automobile type: number of cylinders, displacement, horsepower, weight, acceleration, and model year. Experimental results on this data set were reported in [153]. In the present experiment, instances with missing values were removed, such that the data set contained 392 samples. This set was then divided randomly into training and test subsets, each one containing 196 samples. In [153], *weight* and *model year* were found to be the two most significant input variables. Therefore, these two attributes were used here, in order to keep the presentation simple and to allow for a direct comparison of the obtained results with those reported in [153].

The clustering, performed several times for different cluster numbers, starting with $c = 2$, revealed no significant improvement in the approximation quality when $c > 3$. Therefore, $c = 3$ was selected as the final number of clusters. For each cluster, a set of points with membership grade greater than 0.5 was found. Those points were then used to compute the initial model parameters. The input membership functions obtained by the clustering are shown in Fig. 11 (top). Some positive correlation can be observed between "low weight" and "new model year" and between "average weight" and "more or less new model year."

The prediction error of the initial model was root mean square (RMS) $E_{\text{trn}} = 3.07$ (training data), and RMS $E_{\text{tst}} = 3.09$ (testing data). Tuning the membership functions for 50 epochs, using Levenberg–Marquardt optimization, decreased the RMS errors to $E_{\text{trn}} = 2.91$ and $E_{\text{tst}} = 2.85$, respectively. The prediction accuracy reported in [153] was RMS $E_{\text{trn}} = 2.66$ and $E_{\text{tst}} = 2.97$, using a model with four rules. A fuzzy model with relational rules achieves a comparable accuracy using three rules. Therefore, the relational rule-based fuzzy model compares well with the model reported in [153] in terms of the numerical performance and complexity.

One-dimensional and 2-D membership functions were subsequently extracted from the model. The final 2-D input membership functions are shown in Fig. 11 (bottom) and their 1-D projections are shown in Fig. 12. The following rules provide the linguistic description of the input–input and input–output relationships represented by the data:

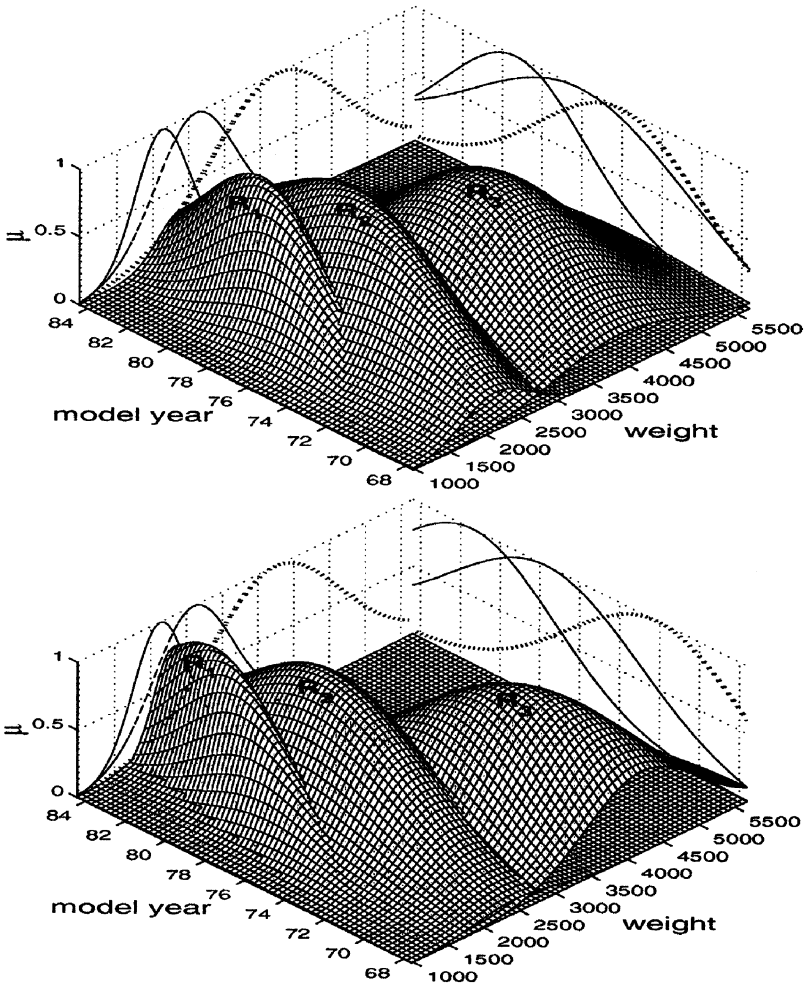

Fig. 11. Initial (top) and final (bottom) 2-D antecedent membership functions for the Auto-MPG example.

$\mathcal{R}_1$: IF ("rather small weight" is moderately positively correlated with "rather new model year") THEN mi/gal is high.

$\mathcal{R}_2$: IF ("small weight" is weakly positively correlated with "more or less new model year") THEN mi/gal is average.

$\mathcal{R}_3$: IF ("more or less large weight" is moderately negatively correlated with "old model year") THEN mi/gal is low.

In terms of the relationships between the output (mi/gal) and the inputs (weight and model year), the conclusion that can be drawn from the three rules above is that miles/gallon correlates positively with model year, i.e., the newer the car, the higher the miles/gallon, and it correlates negatively with the weight, i.e., the heavier the car, the lower the miles/gallon. The three rules divide the universe of the output variable miles/gallon into three fuzzy classes: low, average, and high. The additional knowledge derived from the relational rule antecedents can be explained as follows:

- $\mathcal{R}_1$: The meaning of a moderate positive correlation between "rather small weight" and "rather new model year" is that cars built soon after 1979 with weight slightly more than 2000 lb have the same high miles/gallon as cars built immediately before 1979 with weight slightly less than 2000 miles/gallon.
- $\mathcal{R}_2$: Weak positive correlation can be regarded as a lack of dependency between the weight and model year for cars with medium miles/gallon.
- $\mathcal{R}_3$: Moderate negative correlation between "more or less large weight" and "old model year" means that cars built before 1974 with more than 3700 lb weight have the same low miles/gallon as cars built after 1974, weighing less than 3700 lb.

The numbers used in these explanations are based on the membership functions depicted in Fig. 11 and Fig. 12.

X. Summary, Challenges, and New Directions of Research

Only one approach to data understanding has been reviewed here: the extraction of crisp and fuzzy logical rules from data. Some of the algorithms presented above may also be used to understand what black box classifiers (such as neural networks) really do.

From a geometrical point of view, crisp propositional logic rules provide hyperplanes that divide the feature space per-

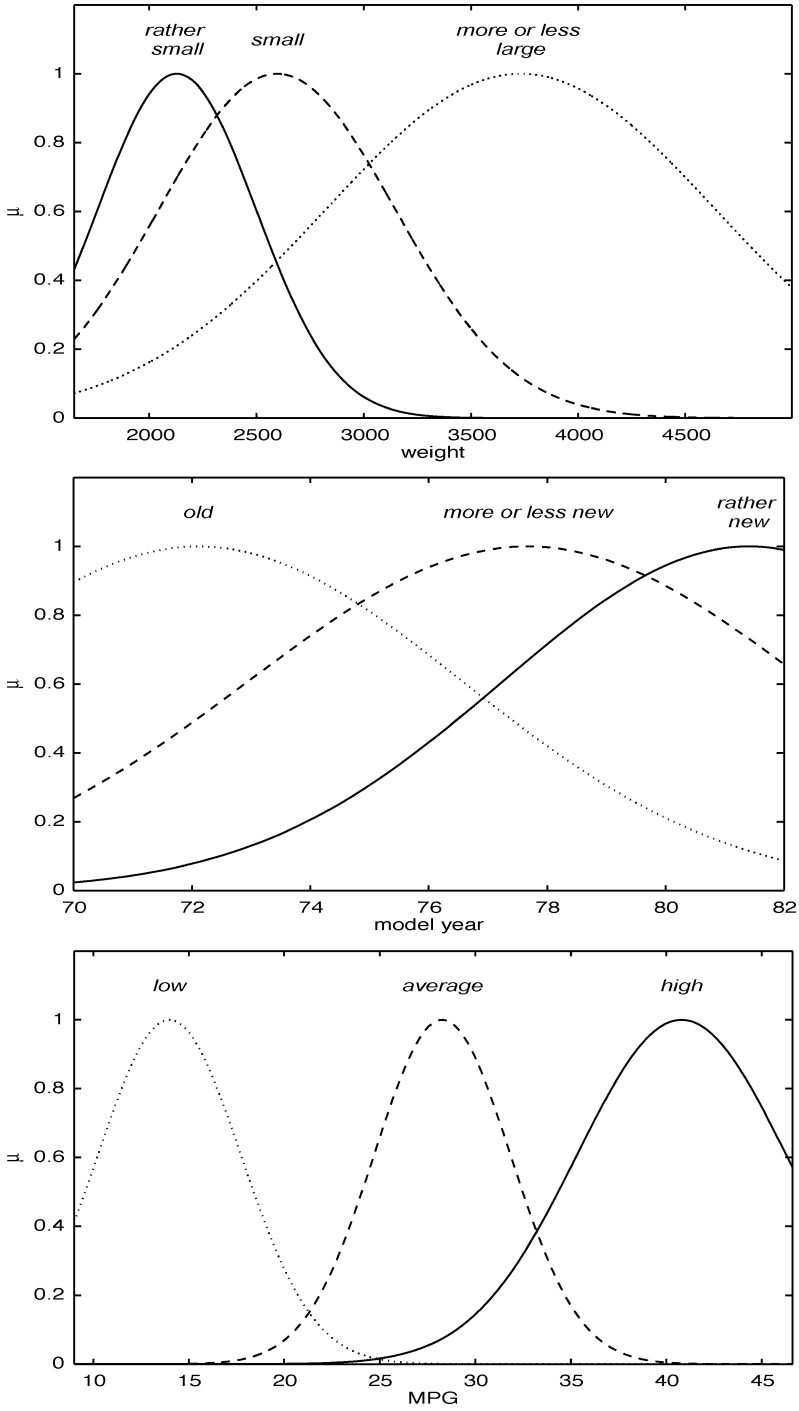

Fig. 12. Final 1-D membership functions for the Auto-MPG example.

pendicular to the axes into areas with symbolic names corresponding to class labels. If the classes in the input space are separated correctly with such hyperplanes, accurate logical description of the data is possible and worthwhile. Otherwise, the logical description of data converges slowly with the number of linguistic variables, leading to complex sets of rules that do not give any insight into the data structure. Fuzzy logic may offer a better approximation with fewer rules, including simple piecewise linear approximation rules and more complex membership functions. However, fuzzy rules based on triangular or Gaussian membership functions provide oval decision borders that do not approximate correctly the sharp decision boundaries that are necessary for describing data with inherent logical structure. Although fuzzy rules are symbolic, their comprehensibility is lower than that of crisp rules. A good strategy is to start with extraction of crisp rules first, try to use fuzzy rules if the results are not satisfactory, and if the number of logical rules is too high or the accuracy of classification is too low, switch to other classification methods.

Logical rules are often highly accurate; they are easy to understand by experts in a given domain, and they may ex-

pose problems with the data itself. Artifacts introduced in real-world datasets by various preprocessing methods may not be obvious if statistical or pattern recognition methods of classification are used. For example, replacing missing features in the data by their averages for a given class leads to good results in cross-validation tests, but for new data samples, of unknown class, the results may be disastrous. The problem may remain obscured by a large number of internal parameters of neural or other classifiers, but logical rules can easily expose it. A medical database that contains several data samples obtained from the same patient may seem easy to classify. In extreme cases there may be no useful information in data, but the nearest-neighbor methods may achieve quite high accuracy, since the training set will contain cases similar to the test cases (Section IX-F). Logical description of such data will be complex and inaccurate, immediately showing that there is a problem.

Although there are many methods that extract logical rules from the data, we have stressed here the advantages of neural networks. The black-box reputation they enjoy is due to their ability to create complex decision borders, but with proper regularization, they may also create decision borders that are equivalent to logical rules. Neural networks are wide-margin classifiers, placing their decision borders in an optimal way, providing linguistic variables without the need for prior discretization, giving sets of rules of different complexity, depending on the regularization parameters, and allowing for the insertion of known rules into the network structure.

After extracting rules, various cost functions for additional optimization of linguistic variables may be used, creating hierarchical sets of logical rules with a different reliability–rejection rate, or a different specificity and sensitivity. A great advantage of fuzzy logic is the soft evaluation of probabilities of different classes, instead of binary yes or no crisp logic answers. Fuzzification of the input values may give the same probabilities as the Monte Carlo procedure performed for input vectors distributed around measured values. Thus, simple interpretation of crisp logical rules is preserved, accuracy is improved by using additional parameters for estimation of measurement uncertainties, and gradient procedures, instead of costly global minimization may be used. Gaussian uncertainties are equivalent to "soft trapezoid" fuzzification of the rectangular crisp membership functions. Sets of crisp logical rules may then be used to calculate probabilities. Novel vectors that would either be rejected or assigned to the default class are assigned to the most probable class.

For several benchmark problems, interesting knowledge in the form of a very simple logical description has been discovered. Logical rules extracted from symbolic data (see our mushroom example) have not been found by human experts before. Although many classifiers have been tried on the Ljubljana cancer data, relatively high accuracy of a rule containing rather trivial knowledge shows that there is no more information in this data. One may argue that the reference accuracy for various problems should be calculated using common sense knowledge, rather than the frequency of the most common class. Sometimes, the simplest logical description of the data may take a form of similarity to a prototype case (see the Wisconsin cancer data).

For some problems, such as the hypothyroid (Section IX-E) or NASA shuttle (Section IX-G), logical rules proved to be more accurate than any other classification method [145], including neural networks. Possible explanations of this empirical observation are the following.

1) The inability of soft transfer functions (sigmoidal or Gaussian) to represent sharp, rectangular boundaries that may be necessary to separate two classes defined by an intrinsic crisp logical rule.
2) The problem of finding a globally optimal solution of the nonlinear optimization problem for neural classifiers; in some cases global optimization method may improve logical rules, but separating optimization of linguistic variables and optimization of rules may still lead to better solutions 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. Bayesian regularization [107], based on weight decay priors [see (13)], improves neural and statistical classification models by smoothing decision borders. It has an adverse effect if sharp decision borders are needed. Sharp decision borders require large weights and thresholds, while regularization terms decrease all weights. Logical rules give much better control over the complexity of the data representation and elimination of outliers—rules that cover only a few new data vectors are identified easily and removed.
4) For medical data, labeling the patients as "sick" or "healthy" introduces implicitly crisp logical rules. Forced to make yes–no diagnoses, human experts may fit the results of tests to specific intervals.

Extracting propositional logical rules from data is relatively simple, but several challenges still remain. The whole process of logical data description and creation of expert systems from extracted rules is still far from being automatic, and perhaps will remain so for a long time. In a way, analysis of each data set is not just a new problem, but quite frequently, it is a new type of problem. In bioinformatics, predictive models are useful, but it is the biological knowledge that has the highest value. Some problems have more than 1000 features and only a few hundred, or even few dozen, samples. If the original features are only weakly correlated with the classes, feature selection methods may not be very helpful. Aggregating input features is done by an MLP2LN neural architecture (Fig. 4), but finding an optimal combination of many input features is difficult, so simpler aggregation methods are needed.

Automatic construction of hierarchical systems that can deal with data containing many missing values is still difficult. Such need arises for example in medicine, where data records frequently contain different features for different patients, since many tests are not performed if initial hypothesis is confirmed. Combining these records into one dataset with common attributes leads to a large number of missing

features. Representation of knowledge in terms of vectors in feature spaces becomes too restrictive.

A vector space representation is also not sufficient to represent dynamic, evolving structural objects [154]. Benchmark and real-world data that require first-order logical rules are not that common. It could be of great importance to formulate clear challenges in these fields and provide more data for empirical tests. Algorithms that treat objects of complex structure already exist [25], [26], [155]. However, there is a balance between generality and efficiency of algorithms for analysis of complex objects, and much more work in this direction is needed to find an optimal balance.

Going beyond propositional logic and simple linguistic variables is of great importance for many applications, such as chemistry, pharmacology, or bioinformatics. One example of such relational rules has been provided here for an approximation problem (mileage prediction). In data mining, relational data are frequently encountered, but extraction of relational logical rules is still not that common. The UCI ML repository [118] has played an important role in the empirical analysis of learning algorithms, but it contains very few relational databases. The lack of simple relational benchmark problems, together with higher difficulty of such problems, contributes to relatively low activity in this area.

Animal brains are very good at making sense data that are important for their survival. An analysis of behavioral patterns, starting from game-playing strategies to analysis of patterns of human interactions requires novel approaches to understanding data. Intelligent robots have to analyze and understand such data, coming as signals from video cameras, microphones and other data sources. Searching for rules in multimedia data requires ingenious filters of relevant information and sophisticated data preprocessing techniques. New types of data are being collected and created with a fast pace. Computational intelligence methods are likely to play an important role in these new fields.

ACKNOWLEDGMENT

The authors would like to thank Dr. A. Gaweda from the School of Medicine, University of Louisville, Louisville, KY, for his contribution to this paper.

REFERENCES

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.
[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
[3] T. Oates and D. Jensen, "Large datasets lead to overly complex models: An explanation and a solution," in *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining*, 1998, pp. 294–298.
[4] R. Andrews, J. Diederich, and A. B. Tickle, "A survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowl.-Based Syst.*, vol. 8, pp. 373–389, 1995.
[5] R. Michalski, "A theory and methodology of inductive learning," *Artif. Intell.*, vol. 20, pp. 111–161, 1983.
[6] I. Roth and V. Bruce, *Perception and Representation*, 2nd ed. Maidenhead, U.K.: Open Univ. Press, 1995.
[7] J. Tukey, *Exploratory Data Analysis*. Menlo Park, CA: Addison-Wesley, 1977.
[8] M. Jambu, *Exploratory and Multivariate Data Analysis*. Boston, MA: Academic, 1991.
[9] W. Duch, "Coloring black boxes: Visualization of neural network decisions," in *Proc. Int. Joint Conf. Neural Networks*, vol. I, 2003, pp. 1735–1740.
[10] M. Jordan and T. J. Sejnowski, Eds., *Graphical Models: Foundations of Neural Computation*. Cambridge, MA: MIT Press, 2001.
[11] 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*, vol. 12, pp. 277–306, Mar. 2001.
[12] T. Fawcett, "Using rule sets to maximize ROC performance," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 131–138.
[13] N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. Cambridge, MA: MIT Press, 1996.
[14] V. Kecman, *Learning and Soft Computing*. Cambridge, MA: MIT Press, 2001.
[15] B. Kosko, *Neural Networks and Fuzzy Systems*. Engelwood Cliffs, NJ: Prentice-Hall, 1992.
[16] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, pp. 338–353, 1965.
[17] T. Bilgiç and I. B. Türkşen, "Measurements of membership functions: Theoretical and empirical work," in *Fundamentals of Fuzzy Sets*, D. Dubois and H. Prade, Eds. Boston, MA: Kluwer, 2000, vol. 1, pp. 195–232.
[18] W. Duch and K. Grudziński, "Prototype based rules—new way to understand the data," in *Proc. Int. Joint Conf. Neural Networks*, 2001, pp. 1858–1863.
[19] Z. Pawlak, *Rough Sets—Theoretical Aspects of Reasoning About Data*. Boston, MA: Kluver, 1991.
[20] S. K. Pal and A. Skowron, *Rough Fuzzy Hybridization: A New Trend in Decision-Making*. New York: Springer-Verlag, 1999.
[21] L. I. Kuncheva, "How good are fuzzy if-then classifiers?," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 501–509, Aug. 2000.
[22] O. Sourina and S. H. Boey, "Geometric query model for scientific and engineering databases," *Int. J. Inf. Technol.*, vol. 2, pp. 41–54, 1996.
[23] W. Duch and G. H. F. Diercksen, "Feature space mapping as a universal adaptive system," *Comput. Phys. Commun.*, vol. 87, pp. 341–371, 1995.
[24] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[25] A. F. Bowers, C. Giraud-Carrier, and J. W. Lloyd, "Classification of individuals with complex structure," in *Proc. 17th Int. Conf. Machine Learning (ICML 2000)*, pp. 81–88.
[26] J. R. Quinlan and R. M. Cameron-Jones, "Induction of logic programs: FOIL and related systems," *New Gener. Comput.*, vol. 13, pp. 287–312, 1995.
[27] W. E. Combs and J. E. Andrews, "Combinatorial rule explosion eliminated by a fuzzy rule configuration," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 1–11, Feb. 1998.
[28] M. K. Güven and K. M. Passino, "Avoiding exponential parameter growth in fuzzy systems," *Trans. IEEE Fuzzy Syst.*, vol. 9, pp. 194–199, Feb. 2001.
[29] W. Duch, R. Adamczak, and K. Grabczewski, "Neural optimization of linguistic variables and membership functions," in *Proc. Int. Conf. Neural Information Processing*, vol. 2, 1999, pp. 616–621.
[30] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. Cambridge, MA: MIT Press, 2001.
[31] D. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," presented at the Machine Learning 12th Int. Conf., San Mateo, CA, 1995.
[32] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Mach. Learn.*, vol. 11, pp. 63–91, 1993.
[33] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous valued attributes for classification learning," presented at the 13th Int. Joint Conf. Artificial Intelligence, San Mateo, CA, 1993.
[34] R. Kohavi and M. Sahami, "Error-based and entropy-based discretization of continuous features," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, 1996, pp. 114–119.
[35] S. Monti and G. F. Cooper, "A latent variable model for multivariate discretization," presented at the Uncertainty 99: The 7th Int. Workshop Artificial Intelligence and Statistics, Fort Lauderdale, FL, 1999.
[36] H. Liu and R. Setiono, "Dimensionality reduction via discretization," *Knowl.-Based Syst.*, vol. 9, pp. 67–72, 1996.
[37] K. Grabczewski and W. Duch, "A general purpose separability criterion for classification systems," in *Proc. 4th Conf. Neural Networks and Their Applications*, 1999, pp. 203–208.
[38] W. Duch, R. Adamczak, and N. Jankowski, "Initialization of adaptive parameters in density networks," in *Proc. 3rd Conf. Neural Networks*, 1997, pp. 99–104.

[39] ——, "New developments in the feature space mapping model," in *Proc. 3rd Conf. Neural Networks*, 1997, pp. 65–70.
[40] W. Duch and N. Jankowski, "A survey of neural transfer functions," *Neural Comput. Surv.*, vol. 2, pp. 163–213, 1999.
[41] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, 1986.
[42] ——, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufman, 1993.
[43] K. Grabczewski and W. Duch, "Heterogenous forests of decision trees," in *Lecture Notes in Computer Science, Artificial Neural Networks*. London, U.K.: Springer-Verlag, 2002, vol. 2415, pp. 504–509.
[44] C. E. Brodley and P. E. Utgoff, "Multivariate decision trees," *Mach. Learn.*, vol. 19, pp. 45–77, 1995.
[45] D. Heath, S. Kasif, and S. Salzberg, "Learning oblique decision trees," in *Proc. 13th Int. Joint Conf. Artificial Intelligence*, 1993, pp. 1002–1007.
[46] S. Murthy, S. Kasif, S. Salzberg, and R. Beigel, "OC1: Randomized induction of oblique decision trees," in *Proc. AAAI Conf.*, 1993, pp. 322–327.
[47] O. L. Mangasarian, R. Setiono, and W. Wolberg, "Pattern recognition via linear programming: Theory and application to medical diagnosis," in *Large-Scale Numerical Optimization*, T. F. Coleman and Y. Li, Eds. Philadelphia, PA: SIAM, 1989, pp. 22–30.
[48] R. Setiono and H. Liu, "A connectionist approach to generating oblique decision trees," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 440–444, June 1999.
[49] L. Breiman, J. H. Friedman, R. A. Oslhen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.
[50] L. Breiman, "Bias-variance, regularization, instability and stabilization," in *Neural Networks and Machine Learning*, C. Bishop, Ed. New York: Springer-Verlag, 1998.
[51] R. S. Michalski, "On the quasiminimal solution of the general covering problem," in *Proc. 5th Int. Symp. Information Processing*, 1969, pp. 125–128.
[52] R. S. Michalski and K. A. Kaufman, "Data mining and knowledge discovery: A review of issues and a multistrategy approach," in *Machine Learning and Data Mining: Methods and Applications*, R. S. Michalski, I. Bratko, and M. Kubat, Eds. New York: Wiley, 1997.
[53] P. Clark and T. Niblett, "The CN2 induction algorithm," *Mach. Learn.*, vol. 3, pp. 261–283, 1988.
[54] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Machine Learning*, 1995, pp. 115–123.
[55] T. M. Mitchell, "Generalization as search," *Artif. Intell.*, vol. 18, pp. 203–226, 1982.
[56] N. Lavrac and S. Dzeroski, *Inductive Logic Programming: Techniques and Applications*. Chichester, U.K.: Ellis Horwood, 1994.
[57] H. Lu, R. Setiono, and H. Liu, "Effective data mining using neural networks," *IEEE Trans. Knowledge Data Eng.*, vol. 8, pp. 957–961, Dec. 1996.
[58] R. Setiono and H. Liu, "Symbolic representation of neural networks," *IEEE Computer*, vol. 29, pp. 71–77, Mar. 1996.
[59] R. Setiono, "Extracting M-of-N rules from trained neural networks," *IEEE Trans. Neural Networks*, vol. 11, pp. 306–512, Mar. 2001.
[60] R. Setiono and W. K. Leow, "FERNN: An algorithm for fast extraction of rules from neural networks," *Appl. Intel.*, vol. 12, pp. 15–25, 2000.
[61] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich, "The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks," *IEEE Trans. Neural Networks*, vol. 9, pp. 1057–1068, Nov. 1998.
[62] K. Saito and R. Nakano, "Medical diagnostic expert system based on PDP model," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, 1988, pp. 255–262.
[63] S. Gallant, *Neural Network Learning and Expert Systems*. Cambridge, MA: MIT Press, 1993.
[64] S. Thrun, "Extracting rules from artifcial neural networks with distributed representations," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MA: MIT Press, 1995.
[65] C. A. Jensen, R. D. Reed, R. J. Marks II, M. A. El-Sharkawi, J.-B. Jung, R. T. Miyamoto, G. M. Anderson, and C. J. Eggen, "Inversion of feedforward neural networks: Algorithms and applications," *Proc. IEEE*, vol. 87, pp. 1536–1549, Sept. 1999.
[66] C. Hernández-Espinosa, M. Fernández Redondo, and M. Ortiz-Gómez, "Inversion of a neural network via interval arithmetic for rule extraction," in *Lecture Notes in Computer Science, Artificial Neural Networks and Neural Information Processing*, 2003, vol. 2714, pp. 670–677.
[67] S. H. Huang and H. Xing, "Extract intelligible and concise fuzzy rules from neural networks," *Fuzzy Sets Syst.*, vol. 132, pp. 233–243, 2002.
[68] L. M. Fu, "Rule learning by searching on adapted nets," in *Proc. 9th Nat. Conf. Artificial Intelligence*, 1991, pp. 590–595.
[69] ——, "Knowledge-based connectionism for revising domain theories," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 173–182, Jan.–Feb. 1993.
[70] ——, *Neural Networks in Computer Intelligence*. New York: McGraw-Hill, 1994.
[71] ——, "Rule generation from neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 1114–1124, Aug. 1994.
[72] R. Setiono, "Extracting rules from neural networks by pruning and hidden-unit splitting," *Neural Comput.*, vol. 9, no. 1, pp. 205–225, 1997.
[73] I. K. Sethi and J. H. Yoo, "Symbolic approximation of feedforward neural networks," in *Pattern Recognition in Practice*, E. S. Gelsema and L. N. Kanal, Eds. New York: North-Holland, 1994, vol. 4.
[74] G. Towell and J. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Mach. Learn.*, vol. 13, pp. 71–101, 1993.
[75] Y. Hayashi, "A neural expert system with automated extraction of fuzzy if-then rules," in *Advances in Neural Information Processing Systems*, R. Lippmann, J. Moody, and D. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, vol. 3, pp. 578–584.
[76] G. Towell and J. Shavlik, "Knowledge-based artificial neural networks," *Artif. Intell.*, vol. 70, pp. 119–165, 1994.
[77] C. McMillan, M. C. Mozer, and P. Smolensky, "Rule induction through integrated symbolic and subsymbolic processing," in *Advances in Neural Information Processing Systems*, J. Moody, S. Hanson, and R. Lippmann, Eds. San Mateo, CA: Morgan Kaufmann, 1992, vol. 4, pp. 969–976.
[78] J. A. Alexander and M. C. Mozer, "Template-based algorithms for connectionist rule extraction," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, vol. 7, pp. 609–616.
[79] R. Setiono and H. Liu, "Understanding neural networks via rule extraction," in *Proc. 14th Int. Joint Conf. Artificial Intelligence*, 1995, pp. 480–485.
[80] M. Ishikawa, "Rule extraction by succesive regularization," in *Proc. 1996 IEEE Int. Conf. Neural Networks*, 1996, pp. 1139–1143.
[81] P. Géczy and S. Usui, "Rule extraction from trained neural networks," *Behaviormetrika*, vol. 26, pp. 89–106, 1999.
[82] W. Duch, R. Adamczak, and K. Grabczewski. (1996) Extraction of logical rules from training data using backpropagation networks. *Proc. 1st Online Workshop Soft Computing* [Online], pp. 25–30. Available: http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/
[83] M. W. Craven and J. W. Shavlik, "Using sampling and queries to extract rules from trained neural networks," in *Proc. 11th Int. Conf. Machine Learning*, 1994, pp. 37–45.
[84] E. Pop, R. Hayward, and J. Diederich, "RULENEG: Extracting rules from a trained ANN by stepwise negation," Neurocomputing Res. Centre, Queensland Univ. Technol., Brisbane, Qld., Aust., QUT NRC Tech. Rep., 1994.
[85] S. Sestito and T. Dillon, *Automated Knowledge Acquisition*. Upper Saddle River, NJ: Prentice-Hall, 1994.
[86] M. J. Healy and T. P. Caudell, "Acquiring rule sets as a product of learning in a logical neural architecture," *IEEE Trans. Neural Networks*, vol. 8, pp. 461–474, May 1997.
[87] A.-H. Tan, "Rule learning and extraction with self-organizing neural networks," in *Proc. 1993 Connectionist Models Summer School*, pp. 192–199.
[88] A. Ultsch, "Knowledge extraction from self-organizing neural networks," in *Information and Classification*, O. Opitz, B. Lausen, and R. Klar, Eds. Berlin, Germany: Springer-Verlag, 1993, pp. 301–306.
[89] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. London, U.K.: Ellis Horwood, 1994.
[90] A. B. Tickle, M. Orlowski, and J. Diederich, "DEDEC: Decision detection by rule extraction from neural networks," Neurocomputing Res. Centre, Queensland Univ. Technol., Brisbane, Qld., Aust., QUT NRC Tech. Rep., 1994.

[91] M. W. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained networks," in *Advances in Neural Information Processing Systems*, D. Touretzky, M. Mozer, and M. Hasselmo, Eds. Cambridge, MA: MIT Press, 1996, vol. 8, pp. 24–30.

[92] J.-S. R. Jang and C. T. Sun, "Functional equivalence between radial basis function neural networks and fuzzy inference systems," *IEEE Trans. Neural Networks*, vol. 4, pp. 156–158, Jan. 1993.

[93] V. Tresp, J. Hollatz, and S. Ahmad, "Network structuring and training using Rule-based knowledge," in *Advances in Neural Information Processing Systems*, J. Moody, S. Hanson, and R. Lippmann, Eds. San Mateo, CA: Morgan Kaufmann, 1993, vol. 4, pp. 871–878.

[94] W. Duch, "Floating Gaussian mapping: A new model of adaptive systems," *Neural Netw. World*, vol. 4, pp. 645–654, 1994.

[95] R. Andrews and S. Geva, "Rules and local function networks," presented at the Rule Extraction From Trained Artificial Neural Networks Workshop, R. Andrews and J. Diederich, Eds., Brighton, U.K., 1996.

[96] ——, "Refining expert knowledge with an artificial neural network," in *Proc. Int. Conf. Neural Information Processing*, vol. 2, 1997, pp. 847–850.

[97] ——, "Rule extraction from a constrained error back propagation MLP," in *Proc. 5th Aust. Conf. Neural Networks*, 1994, pp. 9–12.

[98] P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Zytkow, *Scientific Discovery: Computational Explorations of the Creative Processes*. Cambridge, MA: MIT Press, 1987.

[99] J. J. Mahoney and R. J. Mooney, "Combining neural and symbolic learning to revise probabilistic rule bases," in *Advances in Neural Information Processing Systems*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. San Mateo, CA: Morgan Kaufmann, 1993, vol. 5, pp. 107–114.

[100] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*. Chichester, U.K.: Wiley, 1997.

[101] D. Nauck, U. Nauck, and R. Kruse, "Generating classification rules with the neuro-fuzzy system NEFCLASS," presented at the Biennial Conf. North Amer. Fuzzy Information Processing Soc. (NAFIPS'96), Berkeley, CA.

[102] S. K. Halgamuge and M. Glesner, "Neural networks in designing fuzzy systems for real world applications," *Fuzzy Sets Syst.*, vol. 65, pp. 1–12, 1994.

[103] J. M. Żurada and A. Łozowski, "Generating linguistic rules from data using neuro-fuzzy framework," in *Proc. 4th Int. Conf. Soft Computing*, vol. 2, 1996, pp. 618–621.

[104] H. Surmann and M. Maniadakis, "Learning feed-forward and recurrent fuzzy systems: A genetic approach," *J. Syst. Architect.*, vol. 47, pp. 649–662, 2001.

[105] W. Duch, R. Adamczak, and K. Grabczewski, "Extraction of logical rules from backpropagation networks," *Neural Process. Lett.*, vol. 7, pp. 1–9, 1998.

[106] J. M. Żurada, *Introduction to Artificial Neural Systems*. St. Paul, MN: West, 1992.

[107] D. J. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, pp. 448–472, 1992.

[108] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon Press, 1995.

[109] W. Duch and K. Grabczewski, "Searching for optimal MLP," in *Proc. 4th Conf. Neural Networks and Their Applications*, 1999, pp. 65–70.

[110] M. Kordos and W. Duch, "Multilayer perceptron trained with numerical gradient," in *Proc. Int. Conf. Artificial Neural Networks (ICANN) and Int. Conf. Neural Information Processing (ICONIP)*, 2003, pp. 106–109.

[111] ——, "Search-based training for logical rule extraction by multilayer perceptron," in *Proc. Int. Conf. Artificial Neural Networks (ICANN) and Int. Conf. Neural Information Processing (ICONIP)*, 2003, pp. 86–89.

[112] L. Kanal and V. Kumar, Eds., *Search in Artificial Intelligence*. New York: Springer-Verlag, 1988.

[113] W. Duch, N. Jankowski, K. Grabczewski, and R. Adamczak, "Optimization and interpretation of rule-based classifiers," in *Advances in Soft Computing*. Berlin, Germany: Physica-Verlag, 2000, pp. 1–13.

[114] W. Duch, R. Adamczak, and K. Grabczewski, "Optimization of logical rules derived by neural procedures," presented at the Int. Joint Conf. Neural Networks, Washington, DC, 1999, paper no. 741.

[115] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, pp. 29–36, 1982.

[116] S. K. Pal and S. Mitra, *Neuro-Fuzzy Pattern Recognition*. New York: Wiley, 1999.

[117] R. J. Marks, II, S. Oh, P. Arabshahi, T. P. Caudell, J. J. Choi, and B. J. Song, "Steepest descent adaptation of min-max fuzzy if-then rules," presented at the Int. Joint Conf. Neural Networks, Beijing, China, 1992.

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

[119] D. Nauck, U. Nauck, and R. Kruse, "Generating classification rules with the neuro-fuzzy system NEFCLASS," presented at the Biennial Conf. North Amer. Fuzzy Information Processing Soc. (NAFIPS'96), Berkeley, CA.

[120] S. K. Halgamuge and M. Glesner, "Neural networks in designing fuzzy systems for real world applications," *Fuzzy Sets Syst.*, vol. 65, pp. 1–12, 1994.

[121] C. Browne, I. Düntsch, and G. Gediga, "IRIS revisited: A comparison of discriminant and enhanced rough set data analysis," in *Rough Sets in Knowledge Discovery*, L. Polkowski and A. Skowron, Eds. Heidelberg, Germany: Physica-Verlag, 1998, vol. 2, pp. 345–368.

[122] I. Jagielska, C. Matthews, and T. Whitfort, "The application of neural networks, fuzzy logic, genetic algorithms and rough sets to automated knowledge acquisition," in *Proc. 4th Int. Conf. Soft Computing*, vol. 2, 1996, pp. 565–569.

[123] N. Kasabov, R. Kozma, and W. Duch, "Rule extraction from linguistic rule networks and from fuzzy neural networks: Propositional versus fuzzy rules," in *Proc. 4th Int. Conf. Neural Networks and Their Applications*, 1998, pp. 403–406.

[124] J. Teghem and M. Benjelloun, "Some experiments to compare rough sets theory and ordinal statistical methods," in *Intelligent Decision Support: Handbook of Applications and Advances of Rough Set Theory*, R. Słowiński, Ed. Dordrecht, The Netherlands: Kluwer, 1992, vol. 11, System Theory, Knowledge Engineering and Problem Solving, pp. 267–284.

[125] W. Duch, R. Adamczak, K. Grabczewski, M. Ishikawa, and H. Ueda, "Extraction of crisp logical rules using constrained backpropagation networks—comparison of two new approaches," in *Proc. Eur. Symp. Artificial Neural Networks (ESANN'97)*, pp. 109–114.

[126] S. M. Weiss and C. A. Kulikowski, *Computer Systems That Learn*. San Mateo, CA: Morgan Kauffman, 1990.

[127] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains," in *Proc. 5th Nat. Conf. Artificial Intelligence*, 1986, pp. 1041–1045.

[128] M. Tan and L. Eshelman, "Using weighted networks to represent classification knowledge in noisy domains," in *Proc. 5th Int. Conf. Machine Learning*, 1988, pp. 121–134.

[129] G. Cestnik, I. Konenenko, and I. Bratko, "Assistant-86: A knowledge-elicitation tool for sophisticated users," in *Progress in Machine Learning*, I. Bratko and N. Lavrac, Eds. Wilmslow, U.K.: Sigma, 1987, pp. 31–45.

[130] J. W. Grzymała-Busse and T. Soe, "Inducing simpler rules from reduced data," in *Proc. Workshop Intelligent Information Systems VII*, 1998, pp. 371–378.

[131] F. Zarndt, "A comprehensive case study: An examination of machine learning and connectionist algorithms," M.Sc. Thesis, Dept. Comput. Sci., Brigham Young Univ., Provo, UT, 1995.

[132] K. P. Bennett and O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optim. Methods Softw.*, vol. 1, pp. 23–34, 1992.

[133] R. Setiono, "Generating concise and accurate classification rules for breast cancer diagnosis," *Artif. Intell. Med.*, vol. 18, pp. 205–219, 2000.

[134] B. Ster and A. Dobnikar *et al.*, "Neural networks in medical diagnosis: Comparison with other methods," in *Proc. Int. Conf. EANN'96*, A. Bulsari *et al.*, Eds., pp. 427–430.

[135] N. Shang and L. Breiman, "Distribution based trees are more accurate," in *Proc. Int. Conf. Neural Information Processing*, vol. 1, 1996, pp. 133–138.

[136] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[137] N. Jankowski and V. Kadirkamanathan, "Statistical control of RBF-like networks for classification," in *Proc. 7th Int. Conf. Artificial Neural Networks*, 1997, pp. 385–390.
[138] Ghostminer software, W. Duch, N. Jankowski, K. Grabczewski, A. Naud, and R. Adamczak. [Online]. Available: http://www.fqspl.com.pl/ghostminer/
[139] S. M. Weiss and I. Kapouleas, "An empirical comparison of pattern recognition, neural nets and machine learning classification methods," in *Readings in Machine Learning*, J. W. Shavlik and T. G. Dietterich, Eds. San Mateo, CA: Morgan Kauffman, 1990.
[140] W. Schiffman, M. Joost, and R. Werner, "Comparison of optimized backpropagation algorithms," in *Proc. Eur. Symp. Artificial Neural Networks*, 1993, pp. 97–104.
[141] W. Duch, R. Adamczak, K. Grabczewski, and G. Żal, "Hybrid neural-global minimization method of logical rule extraction," *Int. J. Adv. Comput. Intell.*, vol. 3, pp. 348–356, 1999.
[142] Y. Hayashi, A. Imura, and K. Yoshida, "Fuzzy neural expert system and its application to medical diagnosis," in *Proc. 8th Int. Congr. Cybernetics and Systems*, 1990, pp. 54–61.
[143] S. Mitra, R. De, and S. Pal, "Knowledge based fuzzy MLP for classification and rule generation," *IEEE Trans. Neural Networks*, vol. 8, pp. 1338–1350, Nov. 1997.
[144] Y. Hayashi, R. Setiono, and K. Yoshida, "A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders," *Artif. Intell. Med.*, vol. 20, pp. 205–216, 2000.
[145] W. Duch, R. Adamczak, K. Grabczewski, G. Żal, and Y. Hayashi, "Fuzzy and crisp logical rule extraction methods in application to medical data," in *Fuzzy Systems in Medicine*, P. S. Szczepaniak, P. J. G. Lisboa, and J. Kacprzyk, Eds. Berlin, Germany: Physica-Verlag, 2000, pp. 593–616.
[146] W. Duch, R. Adamczak, and Y. Hayashi, "Neural eliminators and classifiers," in *Proc. 7th Int. Conf. Neural Information Processing (ICONIP 2000)*, S.-Y. Lee, Ed., pp. 1029–1034.
[147] J. N. Butcher and S. V. Rouse, "Personality: Individual differences and clinical assessment," *Annu. Rev. Psychol.*, vol. 47, pp. 87–111, 1996.
[148] W. Duch, R. Adamczak, and K. Grabczewski, "Neural methods for analysis of psychometric data," in *Proc. Int. Conf. Engineering Applications of Neural Networks (EANN'99)*, pp. 45–50.
[149] R. R. Yager and D. Filev, "Relational partitioning of fuzzy rules," *Fuzzy Sets Syst.*, vol. 80, pp. 57–69, 1996.
[150] A. E. Gaweda and J. M. Zurada, "Fuzzy neural network with relational fuzzy rules," in *Proc. Int. Joint Conf. Neural Networks (IJCNN'00)*, vol. 3, pp. 1–2.
[151] ——, "Data-driven linguistic modeling using relational fuzzy rules," *IEEE Trans. Fuzzy Syst.*, vol. 11, pp. 121–134, Feb. 2003.
[152] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.
[153] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft-Computing*. Upper Saddle River, NJ: Prentice-Hall, 1997.
[154] L. Goldfarb, J. Abela, V. C. Bhavsar, and V. N. Kamat, "Can a vector space based learning model discover inductive class generalization in a symbolic environment?," *Pattern Recognit. Lett.*, vol. 16, pp. 719–726, 1995.
[155] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. Neural Networks*, vol. 9, pp. 768–786, Sept. 1998.
[156] R. Hayward, C. Ho-Stuart, J. Diederich, and E. Pop, "RULENEG: Extracting rules from a trained ANN by stepwise negation," Neurocomputing Res. Centre, Queensland Univ. Technol., Brisbane, Qld., Aust., QUT NRC Tech. Rep., 1996.

Włodzisław Duch received the M.Sc. degree in physics, the Ph.D. degree in quantum chemistry, and the D.Sc. degree from the Nicolaus Copernicus University, Toruń, Poland, in 1977, in 1980 in 1986 .

From 1980 to 1982, he was a Postdoctoral Fellow at the University of Southern California, Los Angeles. From 1985 to 1987, he was an Alexander von Humboldt Fellow with the Max Planck Institute of Astrophysics, Munich, Germany. From 1986 to 1997, he was an Associate Professor at the Nicolaus Copernicus University, and in 1997 was granted the title of full Professor. He is currently Head of the Department of Informatics, an independent, interdisciplinary unit within the Faculty of Physics and Astronomy, Nicolaus Copernicus University. He has also been a Visiting Professor in Japan, Canada, Germany, France, and the United States. In 2003, he spent his sabbatical at Nanyang Technological University, Singapore. He has written three books and over 250 papers, and coauthored and edited four books. He is on the Editorial Boards of ten scientific journals. His previous research interests were computational methods of physics and chemistry through foundations of physics. His current research interests are artificial intelligence, neural networks, and cognitive science.

Rudy Setiono (Senior Member, IEEE) received the B.S. degree from Eastern Michigan University, Ypsilanti, in 1984 and the M.Sc and Ph.D. degrees from the University of Wisconsin, Madison, in 1986 and 1990, respectively.

Since 1990, he has been with the National University of Singapore, and he is currently an Associate Professor in the School of Computing. His research interests include linear programming, nonlinear optimization, and neural networks. He is a and

Dr. Setiono is an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS.

Jacek M. Żurada (Fellow, IEEE) is the Samuel T. Fife Alumni Professor of Electrical and Computer Engineering at the University of Louisville, Louisville, KY. He was the Coeditor of *Knowledge-Based Neurocomputing* (Cambridge, MA: MIT Press, 2000), the author of *Introduction to Artificial Neural Systems* (St. Paul, MN: West, 1992), a contributor to the 1994 and 1995 volumes of *Progress in Neural Networks* (Norwood, NJ: Ablex), and Coeditor of *Computational Intelligence: Imitating Life* (Piscataway, NJ: IEEE Press, 1994). He is an Associate Editor of *Neurocomputing.*

Dr. Żurada has received a number of awards, including the 1993 Presidential Award for Research, Scholarship and Creative Activity. He received the University of Louisville President's Distinguished Service Award for Service to the Profession in 2001. In 2003, he was conferred the Title of the Professor by the President of Poland. He is the President of the IEEE Neural Networks Society for 2004–2005 and a Distinguished Speaker of the society. From 1998 to 2003, he was the Editor-in-Chief of IEEE TRANSACTIONS ON NEURAL NETWORKS. He was also an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, Parts I and II, and a Member of the Editorial Board of the PROCEEDINGS OF THE IEEE.