Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

ELSEVIER International Journal of Approximate Reasoning 32 (2003) 103–130

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

Fuzzy data analysis with NEFCLASS

Detlef D. Nauck [*]

BTexact Technologies, Intelligent Systems Lab, Orion Building pp1/12, Adastral Park, Martlesham Heath, Ipswich IP5 3RE, UK

Received 1 January 2002; accepted 1 April 2002

Abstract

Fuzzy data analysis as we interpret it in this paper is the application of fuzzy systems to the analysis of crisp data. In this area, neuro-fuzzy systems play a very prominent role and are applied to a variety of data analysis problems like classification, function approximation or time series prediction. Fuzzy data analysis in general and neuro-fuzzy methods in particular make it easy to strike a balance between accuracy and interpretability. This is an interesting feature for intelligent data analysis and shall be discussed in this paper. We interpret data analysis as a process that is exploratory to some extent. In order for neuro-fuzzy learning to support this aspect we require fast and simple learning algorithms that result in small rule bases, which can be interpreted easily. The goal is to obtain simple intuitive models for interpretation and prediction. We show how the current version of the NEFCLASS structure learning algorithms support this requirement.
© 2002 Elsevier Science Inc. All rights reserved.

Keywords: Fuzzy system; Intelligent data analysis; Neuro-fuzzy methods; Rule learning

1. Introduction

Our modern world is data-driven. Many decisions are made based on the analysis of data. Examples of typical application areas are the weather forecast,

[*] Tel.: +44-1473-605661; fax: +44-1473-642459.
E-mail address: detlef.nauck@bt.com (D.D. Nauck).
URL: http://www.btexact.com/people/nauckd.

stock prediction, the identification of prospective customers, object recognition in images, etc. The remarkable progress in computer technology not only allows us to gather and store more data than we can possibly analyse, it also enables us to do analyses one could not think of 30 years ago. Thus not only the need for data analysis has increased, but also the number of feasible data analysis methods.

The notion of *intelligent* data analysis (IDA) is used to describe a certain approach to data analysis. Like many notions that feature the term *intelligent* IDA also has no exact definition. In our discussion we follow the line of argumentation by Hand found in [7] and [4, Chapter 1].

The two most important areas that contribute to intelligent data analysis are statistics and machine learning. Further contributing areas are soft computing techniques [29] like fuzzy systems, neural networks and probabilistic reasoning – areas that deliberately exploit the tolerance for uncertainty and vagueness in the area of cognitive reasoning. This can lead to considerable reduction in complexity when real-world problems have to be solved and can lead to solutions that are easy to handle, robust, and low-priced. Other areas that can be mentioned are knowledge discovery in databases (KDD), evolutionary computation, artificial intelligence or approximation theory and also database theory that provides means to handle large amounts of data.

From a practical point of view certain restrictions have to be imposed on models obtained in a data analysis process. Thanks to the computer it is possible to create almost arbitrarily sophisticated models that fit every subtle aspect of a data set or an underlying process. Not only are such subtleties usually irrelevant in practical applications, complex models also tend to overfit the data, i.e., they fit the noise and uncertainties contained in the data. From the viewpoint of a user a model must also be comprehensible, interpretable and inexpensive. In several application areas, e.g. medicine or financial services, reasons of security demand that models can only be trusted, if they can be understood by the user. For example an artificial neural network that was created from medical data will probably not be simply accepted as a decision authority, if it recommends an amputation based on the data of a patient. Models obtained from data analysis that are applied in practice usually require transparency and interpretability in terms of the attributes they process. This also requires small models because models with many parameters are not comprehensible to a user.

An important aspect of intelligent data analysis is to select an appropriate model with the application in mind. It may be necessary to sacrifice precision for interpretability, i.e., a suitable balance between model complexity and comprehensibility, between precision and simplicity must be found.

Intelligent data analysis also requires the selection of appropriate algorithms for the process of creating a model. There can be several algorithms available for creating the same kind of model and they may not only differ in computational complexity, speed of convergence, ease of parameterization, but also in the way they ensure certain features in the model they create from data.

Considering the above-given characterisation of IDA we think that fuzzy systems are very valuable for data analysis, especially if simple and interpretable solutions are required [19,20].

The advantage of fuzzy systems is that they can provide simple intuitive models for interpretation and prediction. Prior knowledge in the form of fuzzy rules can be easily integrated. The application of fuzzy systems to data analysis is also known as "fuzzy data analysis". This term refers to the analysis of crisp data with fuzzy methods. There are also approaches that consider the analysis of fuzzy data with generalized statistics [14]. Such approaches are not considered in this paper.

Fuzzy systems conveniently allow us to model a partially known dependency between independent and dependent variables by using linguistic rules. By using linguistic terms represented by fuzzy sets to describe values, we can select a certain granularity under which the data is observed. We can use a fuzzy system both for predicting values for the dependent variables and for knowledge representation.

Thus a fuzzy system can be regarded as a model that links models for prediction and models for understanding. Usually models for prediction are either not interpretable – they are black boxes like, e.g. neural networks – or they are only interpretable by experts – regression models, for example, offer some interpretation of an underlying process. Models for prediction usually do not need to bother with understandability. Their objective is to create accurate predictions.

Models for understanding are usually represented in some kind of rule base, for example, symbolic rules based on predicate calculus. Such rule bases can help to understand an underlying process but building them is often only feasible for finite categorical domains or if numeric domains are partitioned into crisp sets. Such rule bases are less suitable for prediction because counterintuitive results can occur at the boundaries of sets or domains.

Fuzzy systems have numeric interpolation capabilities and are therefore suited for function approximation and prediction. On the other hand they partition variables by fuzzy sets that can be labeled with linguistic terms. Thus they also have a symbolic nature and can be intuitively interpreted. However, there is a trade-off between readability and precision [2]. We can force fuzzy systems to arbitrary precision, but then we lose interpretability. To be very precise, a fuzzy system needs a fine granularity and many fuzzy rules. It is obvious that the larger the rule base of a fuzzy system becomes the less interpretable it gets.

In order to use fuzzy systems in data analysis it must be possible to learn them from examples. Learning in fuzzy systems is most often implemented by learning techniques derived from neural networks. The term *neuro-fuzzy system* (also neuro-fuzzy methods or models) refers to combinations of neural networks and fuzzy systems. This combination does not usually mean that a neural network and a fuzzy system are used together in some way. A neuro-fuzzy method is rather a way to create a fuzzy system from data by some kind of (heuristic) learning method that is motivated by learning procedures used in neural networks.

For IDA we think of neuro-fuzzy methods not mainly as a tool to build predictive models, but as a way to answer questions about data, and for this reason the fuzzy rules that are induced by a neuro-fuzzy method must be comprehensible. In data analysis the emphasis is not on modeling but on answering questions, or as [7] puts it: "it is these questions, not the model per se, which must be paramount".

If a fuzzy model is to be created in such a scenario it is important to apply algorithms that support the exploratory nature of the data analysis process. It is important that the main advantages of a fuzzy system – its simplicity and interpretability – are exploited.

Data analysis also plays an important role in the creation of intelligent systems. From intelligent systems we expect that they can learn, adapt to the users preferences, filter information, act on the behalf of the user, simplify complex information, are simple to use, etc. [1]. Neuro-fuzzy methods can help in achieving some of these goals, especially if we apply neuro-fuzzy methods that focus on interpretability.

In this paper we review NEFCLASS in the context of creating interpretable fuzzy rule bases for data analysis. We have reported about the progresses in the development of NEFCLASS in various publications [20–24]. This paper presents the most recent version of the structure learning algorithm that comprises handling of missing values and symbolic and numeric variables in the same data set. The parameter learning algorithms have not been changed and can be found in one of before-mentioned previous publications.

In the following section we introduce the notation that we use for the rest of this paper and discuss aspects of interpretability of fuzzy systems. In Sections 3–5 we discuss the NEFCLASS system, its structure learning capabilities and its pruning algorithms that are designed to generate small interpretable fuzzy rule bases. Section 6 provides two examples to illustrate the rule learning capabilities before we conclude the paper in Section 7.

2. Interpretable fuzzy systems

If fuzzy sets are used to describe relations between variables we obtain linguistic rules or fuzzy rules. A system of several fuzzy rules is called a fuzzy system. The kind of fuzzy rules that we consider in this paper has the form

$$R_r : \text{if } x_1 \text{ is } \mu_r^{(1)} \text{ and} \ldots \text{and } x_n \text{ is } \mu_r^{(n)} \quad \text{then } y \text{ is } v_r.$$

Usually the fuzzy sets are replaced by their labels, which results in more readable rules like, for instance,

$$R_r : \text{if } x_1 \text{ is } \textit{small} \text{ and} \ldots \text{and } x_n \text{ is } \textit{large} \quad \text{then } y \text{ is } \textit{approximately} \text{ zero}.$$

Other forms of fuzzy rules using additional operators like OR and NOT or systems of hierarchical rules [6] will not be discussed in this paper.

For the fuzzy systems considered in this paper, a fuzzy rule must not be interpreted in the sense of an implication, but as a part of the definition of a function known only at some points. The antecedent describes a vague environment and the consequent provides a vague description of the value that is assumed by the output variable y, if the input vector $(x_1, \ldots, x_n)$ lies within the vague environment described by the antecedent.

A fuzzy system uses a set of such fuzzy rules and provides a computational scheme describing how the rules must be evaluated and combined to compute a crisp output value (vector) for any crisp input vector. One can therefore think of a fuzzy system simply as a parameterized function that maps real vectors to real vectors.

Definition 2.1. A fuzzy system $F_{\mathscr{R}}$ is a mapping

$$F_{\mathscr{R}} : X \to Y,$$

where $X = X_1 \times \cdots \times X_n \subseteq \mathbb{R}^n$ is called a domain or input space, $Y = Y_1 \times \cdots \times Y_m \subseteq \mathbb{R}^m$ is called a co-domain or output space and $\mathbf{x} = (x_1, \ldots, x_n) \in X$ and $\mathbf{y} = (y_1, \ldots, y_m) \in Y$ denote an input vector and an output vector, respectively. $\mathscr{R}$ is a *fuzzy rule* base that determines the structure of the fuzzy system:

$$\mathscr{R} = \{R_1, \ldots, R_r\}.$$

Each rule $R_k \in \mathscr{R}$ is a tuple of fuzzy sets

$$R_k = \left(\mu_k^{(1)}, \ldots, \mu_k^{(n)}, v_k^{(1)}, \ldots, v_k^{(m)}\right),$$

where $\mu_k^{(i)} : X_i \to [0,1]$ is a fuzzy set over the domain of input variable x_i and $v_k^{(j)} : Y_j \to [0,1]$ is a fuzzy set over the domain of output variable y_j. We define

$$F_{\mathscr{R}}(\mathbf{x}) = \mathbf{y} = (y_1, \ldots, y_m),$$

where

$$y_j = \text{defuzz}\Big(\perp_{R_k \in \mathscr{R}} \left\{\hat{v}_k^{(j)}\right\}\Big) \quad \text{with}$$

$$\hat{v}_k^{(j)} : Y_j \to [0,1], \;\; y_j \mapsto \top_2\left\{\tau_k, v_k^{(j)}\right\} \quad \text{with}$$

$$\tau_k = \top_1\left\{\mu_k^{(1)}(x_1), \ldots, \mu_k^{(n)}(x_n)\right\},$$

where $\top_1$ and $\top_2$ are t-norms, $\perp$ is a t-conorm, τ_k is the degree fulfilment of fuzzy rule R_k and defuzz is a defuzzification method that is used to convert an output fuzzy set $\hat{v}_k^{(j)}$ into a crisp output value.

Fuzzy systems as they are introduced by the previous definition are linguistic representations of piecewise defined functions. The evaluation of the rule base provides an interpolation strategy in a vague environment. The inputs and outputs are crisp values. Only the internal computation is fuzzy.

Fuzzy systems can also be used for classification problems which can be interpreted as a special case of function approximation. In a crisp classification problem an input vector (pattern) must be assigned to one of several classes. A class is a subset of the pattern space. A fuzzy classification problem accepts that a pattern is assigned to several classes with different degrees of membership. In this case a class is a fuzzy set of the pattern space. A classification problem can be easily be transformed into a function approximation problem by specifying a set $\tilde{\mathscr{L}}$ with patterns $(\mathbf{x}, \mathbf{c})$, where $\mathbf{c} \in [0,1]^m$ and c_i denotes the degree of membership of $\mathbf{x}$ in class C_i. If a crisp classification problem is represented this way, then in each $\mathbf{c}$ there is exactly one component set to 1 and all other components are set to 0. A fuzzy system used for classification is called a fuzzy classifier.

Definition 2.2. A *fuzzy classifier* is a fuzzy system

$$F_{\mathscr{R}} : X \to Y$$

with $Y = [0,1]^m$. Its rule base $\mathscr{R}$ consists of special kinds of fuzzy rules of the form

$$R_k = \left(\mu_k^{(1)}, \ldots, \mu_k^{(n)}, c_{j_k}\right),$$

where $c_{j_k} \in C = \{c_1, \ldots, c_m\}$ is a class label. We define

$$F_{\mathscr{R}}(\mathbf{x}) = \mathbf{y} = (y_1, \ldots, y_m)$$

with

$$y_j = \mathop{\perp}_{\substack{R_k \in \mathscr{R} \\ \mathrm{con}(R_k) = c_j}} \{\tau_k\},$$

where $\perp$ is a t-conorm and $\mathrm{con}(R_k)$ is the consequent of rule R_k.

The output of a fuzzy classifier is a vector whose components denote the degree of membership of a processed pattern to the available classes. In many applications a pattern must be assigned to a single class only. In this case the output vector of a fuzzy classifier must be interpreted (or defuzzified). Usually a "winner takes all" interpretation is used, i.e., the class with the largest degree of membership is selected.

We interpret fuzzy systems as convenient models to linguistically represent (non-linear) mappings [30]. The designer of a fuzzy system specifies characteristic points of an assumed underlying function. This function is unknown except for those characteristic points. The fuzzy sets that are used to linguistically describe those points express the degree of indistinguishability of points that are close to each other. Fuzzy systems can be interpreted on the basis of equality relations [12,13].

The advantages of applying a fuzzy system are the simplicity and the linguistic interpretation of the approach. This allows for the inexpensive and fast development and maintenance of solutions and thus enables us to solve problems in application areas where rigorous formal analysis would be too expensive and time-consuming.

In the area of data analysis the interpretability and simplicity of fuzzy systems are the key advantage. Fuzzy systems are not better function approximators or classifiers than other approaches. If we want to keep the model simple, the prediction is usually less accurate. This means fuzzy systems should be used for data analysis, if an interpretable model is needed that can also be used to some extent for prediction.

Interpretability of a fuzzy model should not mean that there is an exact match between the linguistic description of the model and the model parameters. This is not possible anyway, due to the subjective nature of fuzzy sets and linguistic terms. Interpretability means that the users of the model can accept the representation of the linguistic terms, more or less. The representation must roughly correspond to their intuitive understanding of the linguistic terms. It is more important that the rule base is small and thus comprehensible.

Furthermore, interpretability should not mean that *anybody* can understand a fuzzy system. It means that users who are at least to some degree experts in the domain where the data analysis takes place can understand the model. Obviously we cannot expect a lay person to understand a fuzzy system in a medical domain. It is important that the medical expert who uses the model should understand it.

From the viewpoint of a user we can formulate the following intuitive criterion for the interpretability of a fuzzy system. We assume that the linguistic interpretability of a fuzzy system is adequate if

- it provides a rough idea about the underlying process or the relations within the data,
- it sufficiently justifies the majority of observed output values,
- it is usable for explanations,
- it covers all important observed input/output situations (rare cases or exceptions might be ignored).

A neuro-fuzzy learning procedure for creating interpretable fuzzy systems in data analysis must be simple and fast to allow a user to understand what it does and to experiment with it. We prefer a tool-oriented, exploratory view on neuro-fuzzy systems. The learning algorithm should take the semantics of the desired fuzzy system into account, and adhere to certain constraints. The learning result should also be interpreted, and the insights gained by this should be used to restart the learning procedure to obtain better results if necessary. A neuro-fuzzy system supports the user in finding a desired fuzzy system based on training data, but it cannot do all the work. This view matches the exploratory nature of intelligent data analysis.

Semantical problems will occur if neuro-fuzzy systems do not have mechanisms to make sure that all changes caused by the learning procedure are interpretable in terms of a fuzzy system [18]. The learning algorithms should be constrained such that adjacent membership functions do not exchange positions, do not move from positive to negative parts of the domains or vice versa, have a certain degree of overlapping, etc. An interpretation in terms of a Mamdani-type fuzzy system may not be possible if the evaluation of antecedents is not done by t-norms, but by certain special functions. The following points influence the interpretability of a fuzzy system:

- The number of fuzzy rules: a fuzzy system with a large rule base is less interpretable than a fuzzy system that needs only few rules.
- The number of variables: high dimensional models are incomprehensible. Each rule should use as few variables as possible.
- The number of fuzzy sets per variable: only a few meaningful fuzzy sets should be used to partition a variable. A fine granularity not only increases the number of linguistic terms for a variable, but also the number of possible fuzzy rules increases exponentially with the number of variables and fuzzy sets. A coarse granularity increases the readability of the fuzzy model.
- Unambiguous representation of linguistic terms: each linguistic term must be represented by only one fuzzy set. Different rules using the same linguistic expression (e.g. x is *small*) may not represent the corresponding linguistic term (e.g. *small*) by different fuzzy sets.
- No conflicts: there must be no rules in the rule base that have identical antecedents but different consequents (complete contradiction). Only partial contradiction is acceptable.
- No redundancy: no rule may appear more than once in the rule base. There must also be no rule whose antecedent is a subset of the antecedent of another rule.
- Characteristics of fuzzy sets: fuzzy sets should be "meaningful" to the user of the fuzzy system. After training, the fuzzy partition of a variable should still be reasonably similar to the partition provided by the user. At least the relative position of the fuzzy sets must be maintained. Usually, a minimum/

maximum degree of overlapping must be enforced. Fuzzy sets should be normal and convex and be interpretable as fuzzy numbers or fuzzy intervals (for numeric variables, symbolic variables are discussed, e.g. in [17]).

Aspects of interpretability can become more complex, if different types of fuzzy rule bases are considered, for example hierarchical rule bases [6]. For a discussion about the trade-off between accuracy and interpretability, see also [3,5].

3. NEFCLASS

The idea of a neuro-fuzzy system is to find the parameters of a fuzzy system by means of learning methods obtained from neural networks. Learning in fuzzy systems must consider structure learning, i.e., creation of a rule base, and parameter learning, i.e., optimization of fuzzy sets. Parameter learning is often done by algorithms that were inspired by neural network learning. Structure learning on the other hand is usually not taken from neural networks. The term "neuro-fuzzy", however, is nowadays applied to almost all approaches to learning in fuzzy systems such that the learning of fuzzy rules is also subsumed under this notion [8,20]. Distinctions are only made, for example, when fuzzy rules are created by fuzzy decision tree learning [9] or by genetic algorithms [11,15].

A common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural-network-like architecture. Then a learning algorithm – such as backpropagation – is used to train the system. There are some problems, however. Neural network learning algorithms are usually based on gradient descent methods. They cannot be applied directly to a fuzzy system, because the functions used in the inference process are usually not differentiable. There are two solutions to this problem: either replace the functions used in the fuzzy system (like min and max) by differentiable functions, or do not use a gradient-based neural learning algorithm but a better-suited procedure. The NEFCLASS system that we discuss here uses the latter approach.

There are several different approaches which have much in common, but differ in implementational aspects. To stress the common features of all these approaches, and to give the term *neuro-fuzzy system* a suitable meaning, we only apply it to systems which possess the following properties:

(i) A neuro-fuzzy system is a fuzzy system that is trained by a learning algorithm (usually) derived from neural network theory. The (heuristic) learning procedure operates on local information, and causes only local modifications in the underlying fuzzy system. The learning process is not knowledge-based, but data-driven.

(ii) A neuro-fuzzy system can always (i.e., before, during and after learning) be interpreted as a system of fuzzy rules. It is possible both to create the system out of training data from scratch, and to initialize it from prior knowledge in the form of fuzzy rules.
(iii) The learning procedure of a neuro-fuzzy system takes the semantical properties of the underlying fuzzy system into account. This results in constraints on the possible modifications of the system's parameters.
(iv) A neuro-fuzzy system approximates an n-dimensional (unknown) function that is partially given by the training data. The fuzzy rules encoded within the system represent vague samples, and represent vague prototypes of the training data. A neuro-fuzzy system should not be seen as a kind of (fuzzy) expert system, and it has nothing to do with fuzzy logic in the narrow sense [13].
(v) A neuro-fuzzy system can be represented by a special three-layer feedforward neural network (see Definition 3.1). This view of a fuzzy system illustrates the data flow within the system and its parallel nature. However, this neural network view is not a prerequisite for applying a learning procedure, it is merely a convenience.

The neuro-fuzzy technique, then, is used to derive a fuzzy system from data, or to enhance it by learning from examples. The exact implementation of the neuro-fuzzy model does not matter. A lot of neuro-fuzzy approaches use a neural network-like graph to illustrate the data flow and the computations that are carried out in a fuzzy system. This neural network representation is then used to formalize the application of a learning algorithm.

A 3-*layer fuzzy perceptron* provides a way to represent a fuzzy system as a network. The name refers to the structure of the model that is similar to a multilayer perceptron, but where the weights are modeled as fuzzy sets and the activation, output, and propagation functions are changed accordingly, to implement a common fuzzy inference path. The term *fuzzy* (*multilayer*) *perceptron* has also been used by other authors for their approaches [10,16,26]. Here this notion is used to describe the topology.

Definition 3.1. Let $F_{\mathscr{R}}$ be a fuzzy system. A *3-layer fuzzy perceptron* is a network representation of a fuzzy system $F_{\mathscr{R}}$ in the form of a neural network $(U, W, A, O, \mathrm{NET}, \mathrm{ex})$, where
(i) $U = U_1 \cup U_2 \cup U_3$ with $U_1 = \{x_1, \ldots, x_n\}$, $U_2 = \{R_1, \ldots, R_r\}$, $U_3 = \{y_1, \ldots, y_m\}$.
(ii) W, the network structure, is a partial mapping from $U \times U \to \mathscr{F}(\mathbb{R})$ and is given by

$$W(u,v) = \begin{cases} \mu_j^{(i)} & \text{if } u = x_i,\ v = R_j, \\ \nu_j^{(k)} & \text{if } u = R_j,\ v = y_k \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant r$ and $1 \leqslant k \leqslant m$. In addition every two connections with weights $W(u,v)$ and $W(u',v')$ become coupled connections, if $W(u,v) = W(u',v')$ and $(u = u', u, u' \in U_1 \wedge v \neq v', v, v' \in U_2)$ or $(u \neq u', u, u' \in U_2 \wedge v = v', v, v' \in U_3)$ holds.

If $W(u,v)$ and $W(u',v')$ are coupled, then if $W(u,v)$ is modified by a learning algorithm, $W(u',v')$ is modified in the same way and vice versa.

(iii) A is a mapping that assigns an activation function A_u to each $u \in U$ with

$$A_u : \mathbb{R} \to \mathbb{R}, \quad a_u = A_u(\mathrm{net}_u) = \mathrm{net}_u \quad \text{for } u \in U_1 \cup U_2,$$

and

$$A_u : \mathscr{F}(\mathbb{R}) \to \mathscr{F}(\mathbb{R}), \quad a_u = A_u(\mathrm{net}_u) = \mathrm{net}_u \quad \text{for } u \in U_3.$$

(iv) O is a mapping that assigns an output function O_u to each $u \in U$ with

$$O_u : \mathbb{R} \to \mathbb{R}, \quad o_u = O_u(a_u) = a_u \quad \text{for } u \in U_1 \cup U_2,$$

and

$$O_u : \mathscr{F}(\mathbb{R}) \to \mathbb{R}, \quad o_u = O_u(a_u) = \mathrm{defuzz}(a_u) \quad \text{for } u \in U_3.$$

(v) NET is a mapping that assigns a network input function NET_u to each $u \in U$, with

$$\mathrm{NET}_u : \mathbb{R} \to \mathbb{R}, \quad \mathrm{net}_u = \mathrm{NET}_u(\mathrm{ex}_u) = \mathrm{ex}_u \quad \text{for } u \in U_1,$$

$$\mathrm{NET}_u : (\mathbb{R} \times \mathscr{F}(\mathbb{R}))^{U_1} \to [0,1], \quad \mathrm{net}_u = \mathop{\top_1}_{u' \in U_1} \{W(u',u)(o_{u'})\} \quad \text{for } u \in U_2$$

and

$$\mathrm{NET}_u : ([0,1] \times \mathscr{F}(\mathbb{R}))^{U_2} \to \mathscr{F}(\mathbb{R}), \quad \mathrm{net}_u : \mathbb{R} \to [0,1],$$

$$\mathrm{net}_u(y) = \mathop{\perp}_{u' \in U_2} \{\top_2\{o_{u'}, W(u',u)(y)\}\} \quad \text{for } u \in U_3.$$

(vi) $\mathrm{ex} : U_1 \to \mathbb{R}$ defines for each input unit $u \in U_1$ its external input $\mathrm{ex}(u) = \mathrm{ex}_u$. For all other units ex is not defined.

In [20] we find a definition for a *generic* fuzzy perceptron that is more general than the fuzzy perceptron of Definition 3.1, as it does not enforce coupled connections. A generic fuzzy perceptron can be used to derive neuro-fuzzy models for special domains, and can serve as a common foundation to evaluate different neuro-fuzzy approaches by means of the same underlying model.

In this paper we are interested in creating an interpretable fuzzy system for data analysis. One important feature of interpretable fuzzy systems is that no linguistic expression is represented by more than one fuzzy set. Therefore we must take care that a network representation uses coupled connections (shared weights), as required in Definition 3.1.

NEFCLASS (neuro-fuzzy classification) is a neuro-fuzzy approach to derive fuzzy classification rules from a set of labelled data [21–23]. NEFCLASS creates

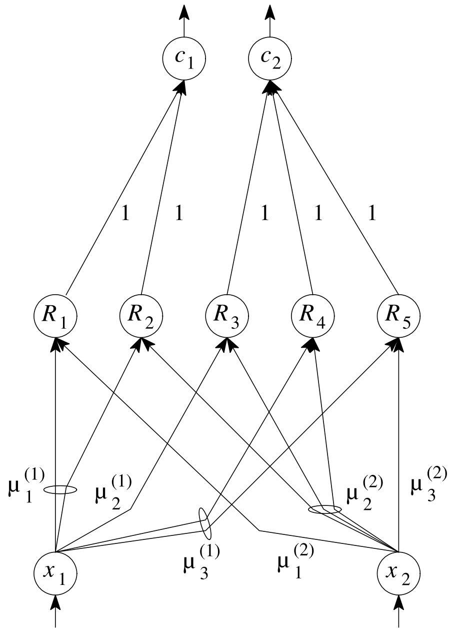

Fig. 1. A network representation of a NEFCLASS system.

a fuzzy classifier according to Definition 2.2 by using the structure-oriented rule learning algorithms (see following section). A NEFCLASS system (Fig. 1) can be represented as a special kind of fuzzy perceptron, because Definition 3.1 is flexible enough to allow for the representation of fuzzy classifiers.

Definition 3.2. A NEFCLASS system represents a fuzzy classifier $F_{\mathscr{R}}$ according to Definition 2.2 with a set of class labels $C = \{c_1, \dots, c_m\}$, $\top_1 = \min$, $\top_2 = \min$ and $\perp = \max$. A network representation of a NEFCLASS system is a fuzzy perceptron according to Definition 3.1 with the following specifications:

(i) W, the network structure, is a partial mapping from $U \times U \to \mathscr{F}(\mathbb{R})$ and is given by

$$W(u,v) = \begin{cases} \mu_j^{(i)} & \text{if } u = x_i (i \in \{1,\dots,n\}) \wedge v = R_j (j \in \{1,\dots,r\}), \\ \mathbb{I}_{\{1\}} & \text{if } u = R_j \wedge v = c_{i_j} = \text{con}(R_j) \\ & (j \in \{1,\dots,r\})(i_j \in \{1,\dots,m\}), \\ \text{undefined} & \text{otherwise} \end{cases}$$

In addition every two connections with weights $W(u,v)$ and $W(u',v')$ $(u = u', v \neq v', u, u' \in U_1, v, v' \in U_2)$ become coupled connections, if $W(u,v) = W(u',v')$ holds.

(ii) The network input for the third layer is computed as follows:

$$\mathrm{NET}_u : ([0,1] \times \mathscr{F}(\mathbb{R}))^{U_2} \to \mathscr{F}(\mathbb{R}), \quad \mathrm{net}_u : \mathbb{R} \to [0,1],$$
$$\mathrm{net}_u(y) = \max_{u' \in U_2} \{ \min\{o_{u'}, W(u',u)(y)\}\}$$

for $u \in U_3$.

(iii) The output of a unit in the third layer is given by

$$O_u : \mathscr{F}(\mathbb{R}) \to \mathbb{R}, \quad o_u = O_u(a_u) = \mathrm{defuzz}(a_u) = \mathrm{height}(a_u)$$

for $u \in U_3$.

In Fig. 1 a NEFCLASS system with two inputs, five rules and two classes is shown. The main difference between a fuzzy perceptron and a NEFCLASS system is that only one connection protrudes from each unit of the second layer to one unit of the third layer. This connection represents the connection between a rule unit and the class used in the consequent of the corresponding rule. These connections have a constant weight of 1, which actually means they are not weighted. Weighted fuzzy rules are difficult to interpret. This problem of weighted fuzzy rules is discussed in [18]. In order to keep the structure of a fuzzy perceptron that demands a fuzzy set attached to these connections we use the membership function $\mathbb{I}_{\{1\}}$ – which is the characteristic function of the set $\{1\}$ – to represent the singleton 1 as a fuzzy set (Definition 3.2(i)).

An output unit receives a modified version of this fuzzy set, i.e., its height is reduced to the maximum of the output values of all rule units connected to the considered output unit (Definition 3.2(ii)). The output values of the rule units are the degrees of fulfilment of the corresponding fuzzy rules. The output unit then defuzzifies this output fuzzy set by means of a special defuzzification function that computes the height of the output fuzzy set (Definition 3.2(iii)).

Because NEFCLASS uses coupled connections (shared weights), for each linguistic value there is only one representation as a fuzzy set. This ensures the interpretability of the fuzzy rule base. During learning it cannot happen that two fuzzy sets corresponding to the same label (e.g. *positive big*) develop differently. In Fig. 1 shared weights are denoted by ellipses around the connections. Connections that share a weight always come from the same input unit.

4. NEFCLASS structure learning

In order to be useful as a data analysis tool for classification problems, the NEFCLASS implementation provides following features:

- fast generation of fuzzy classifiers through simple learning strategies,
- constrained fuzzy set learning to retain the interpretability of a generated classifier,
- automatic pruning to reduce the complexity of a generated classifier,

- automatic cross-validation to generate error estimates for a generated classifier,
- methods for integrating prior knowledge and for modifying generated classifiers manually,
- treatment of missing values,
- both numeric and symbolic variables,
- treatment of unbalanced data sets.

In this section we present the most recent structure learning algorithm for NEFCLASS. The algorithm can handle missing values and any mixture of numeric and symbolic data. The primary rule learning procedure is given in Algorithm 1. It computes a fuzzy rule base by using a modified Wang–Mendel procedure [27] to find antecedents. If a numeric feature is missing, it results in more rules to be created, because all available fuzzy sets for this variable, are eligible for the antecedent. After the numeric variables have been processed, the antecedents are completed by adding fuzzy sets for the symbolical features. Those fuzzy sets are not parameterised, but they store the membership degrees individually for each attribute value. A fuzzy set for a symbolic feature is therefore basically a normalise histogram. If a value of a symbolic feature is missing, no further action is required.

After the antecedents have been created, we select for each antecedent a suitable consequent, such that the resulting rule cause as few errors as possible. To obtain the final rule base, we select an appropriate number from the list of created rules. This number can either be specified by the user or computed automatically such that all patterns are covered by rules. Algorithm 2 shows a rule selection procedure that tries to create an equal number of rules for each class.

The algorithms use the following notations:

- $\tilde{\mathscr{L}}$: a set of training data (fixed learning problem) with $|\tilde{\mathscr{L}}| = s$, which represents a classification problem where patterns $\mathbf{p}$ are to be assigned to m classes $C_1, \ldots, C_m$.
- $(\mathbf{p}, \mathbf{t}) \in \tilde{\mathscr{L}}$: a training pattern consists of an input vector $\mathbf{p} \in X_1 \times \cdots \times X_n$ and a target vector $\mathbf{t} \in [0,1]^m$. $\mathbf{p}$ consists of u numeric and v symbolic features ($u + v = n$), i.e., X_i is either a subset of $\mathbb{R}$ or a (finite) set of symbols. The target vector represents a possibly vague classification of the input pattern $\mathbf{p}$. The class index of $\mathbf{p}$ is given by the index of the largest component of $\mathbf{t}$: $\text{class}(\mathbf{p}) = \text{argmax}_j\{t_j\}$.
- $R = ((A, M), C)$: a fuzzy classification rule with antecedent $\text{ant}(R) = (A, M)$ and consequent $\text{con}(R) = C$, which denotes a class. $A = (\mu_{j_1}^{(1)}, \ldots, \mu_{j_u}^{(u)})$ is created by fuzzy sets for numeric variables and is the first part of the antecedent. $M = (\mathbf{m}_{j_1}^{(1)}, \ldots, \mathbf{m}_{j_v}^{(v)})$ is the second part of the antecedent and is created by the fuzzy sets for the symbolic variables.
- $\mu_j^{(i)}$: jth fuzzy set of the fuzzy partition of input variable x_i. There are q_i fuzzy sets for variable x_i.

- $\mathbf{m}_j^{(k)}$: jth fuzzy set of the kth symbolic variable x_k. There are m fuzzy sets for each symbolic variable, i.e., one fuzzy set per class. A fuzzy set $\mathbf{m}_j^{(k)}$ is represented by a vector that contains the degrees of membership for all elements of X_k. At the time of initialization (Algorithm 1, line 12) all entries of $\mathbf{m}_j^{(k)}$ are set to zero. We use $\mathbf{m}_j^{(k)}[x]$ to denote that the degree of membership for x is accessed for manipulation (Algorithm 1, line 22).
- $P_R \in [-1, 1]$: a value representing the performance of rule R:

$$P_R = \frac{1}{s} \sum_{(\mathbf{p},\mathbf{t})\in\tilde{\mathscr{L}}} (-1)^c R(\mathbf{p}) \quad \text{with } c = \begin{cases} 0 & \text{if class}(\mathbf{p}) = \text{con}(R), \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

Algorithm 1 starts by creating initial antecedents that contain only numeric attributes using a Wang–Mendel procedure [27]. After the training data is processed once, all k antecedents that are supported by the data have been found. Then from each antecedent m rules are created, one for each class, and the initial antecedents are completed by constructing fuzzy sets for the symbolic attributes by counting the frequencies of the symbolic values [17]. This means there is now an initial rule base that contains a set of $m \cdot k$ rules. This rule set can be inconsistent, because it can contain contradictory rules. After resolving inconsistencies, by selecting the rule with a better performance from multiple rules with identical antecedents but different consequents, a final list of rule base candidates is created. Then a rule evaluation algorithm is applied to select a final rule base [19]. One example is given in Algorithm 2.

After rule creation the fuzzy sets of both numeric and symbolic variables are trained to improve the performance of the classifier [17,19,20].

The mixed fuzzy rules created by Algorithm 1 cannot be as easily interpreted as fuzzy rules that use only numeric variables and continuous membership functions, which can be labelled with terms like *small* or *large*.

Fuzzy sets that are denoted as an ordered list of pairs are hard to be labelled linguistically. In some cases linguistic labels can be found by inspection. For example, if we have a symbolic variable describing the job of a person the fuzzy set {(accountant, 0), (consultant, 0.3), (engineer, 0.7), (lecturer, 1), (professor, 1)} may be labelled by *academic job*.

If fuzzy rules are created by learning, then it is useful to also create linguistic labels automatically. To quickly generate a rough linguistic term for a fuzzy set given by an ordered list of pairs we could use "y is A or C or B" for y is $\{(A, 1.0), (B, 0.4), (C, 0.7)\}$. The order in which the feature values with non-zero membership are listed, expresses the preferences represented in the degrees of membership. In this case we learn from the label, that A is more typical than C and C is more typical than B. If we need to know the exact degrees of membership, we can look at the fuzzy set.

This interpretation is similar to common linguistic labels like *approximately zero* for a numeric variable. In this case we also know, that 0 is the most typical value for the variable and larger or smaller values are less typical. If we are interested in the exact degrees, we also have to look at the membership function.

Algorithm 1 (*NEFCLASS structure learning*).

```
1: for all $(\mathbf{p},\mathbf{t}) \in \tilde{\mathscr{L}}$ do
2:                                                      (* find all hyperboxes that contain data *)
3:   create a new empty antecedent $A'$;
4:   for all metric input features $x_i$ whose value is not missing do
5:     $\mu_{j_i}^{(i)} = \mathrm{argmax}_{\mu_j^{(i)}, j\in\{1,\ldots,q_i\}}\{\mu_j^{(i)}(p_i)\}$;
6:     add $\mu_{j_i}^{(i)}$ to antecedent $A'$;
7:   end for
8:   repeat
9:     create a new empty antecedent $A$;
10:      $A = A'$;
11:      create a new combination of fuzzy sets from all missing input features and add them to $A$;
12:      if ($A \notin$ list of antecedents) then
13:        add antecedent $A$ to list of antecedents;
14:      end if
15:    until all combinations were enumerated
16: end for
17: for all $A \in$ list of antecedents do
18:                                                     (* create rule base candidates *)
19:    initialize $M$;
20:    create complete antecedent $(A, M)$;
21:    for all classes $C$ do
22:      create rule $R = ((A, M), C)$ and add it to list of rule base candidates;
23:    end for
24: end for
25: for all $(\mathbf{p},\mathbf{t}) \in \tilde{\mathscr{L}}$ do
26:                                                     (* compute frequencies of symbolic variables *)
27:    for all $R \in$ list of rule base candidates do
28:      if (class($\mathbf{p}$) = con($R$)) then
```

```
29:         for all symbolic features $x_k$ do
30:           if($x_k$ is not missing) then
31:             with $R$ do: $\mathbf{m}^{(k)}[p_k] = \mathbf{m}^{(k)}[p_k] + 1$;
32:           end if
33:         end for
34:       end if
35:     end for
36: end for

37: for all $R \in$ list of rule base candidates do
38:     with $R$ do: normalize all $\mathbf{m}^{(i)}$;
39:                                             (* transform the $\mathbf{m}^{(i)}$
                                                into fuzzy sets *)
40:     compute performance $P_R$ of $R$;       (* see Eq. (1)*)
41: end for

42: Find all contradicting rules and resolve con-
    flicts;
43: if (select best rules) then
44:     SelectBestRules;                        (* see [19]*)
45: else if (select best rules per class) then
46:     SelectBestRulesPerClass;                (* see Algorithm 2 *)
47: end if
```

Algorithm 2 (*Select best rules per class*).

SelectBestRulesPerClass

```
1: $k = 0$; stop = false;
2: repeat
3:   for all classes $C$ do
4:     if ($\exists R : \mathrm{con}(R) = C$) then
5:       $R' = \underset{R:\mathrm{con}(R)=C}{\mathrm{argmax}}\{P_R\}$;
6:       if (fixed rule base size) then
7:         if ($k < k_{\max}$) then
8:           add $R'$ to rule base;
9:           delete $R'$ from list of rule candidates;
10:          $k = k + 1$;
11:        else
12:          stop = true;
13:        end if
14:      else if (all patterns must be covered) then
15:        if ($R'$ covers some still uncovered patterns) then
```

```
16:             add $R'$ to rule base;
17:             delete $R'$ from list of rule candidates;
18:           **end if**
19:           **if** (all patterns are now covered) **then**
20:             stop = true;
21:           **end if**
22:         **end if**
23:       **end if**
24:     **end for**
25: **until** stop
```

5. Pruning fuzzy rule bases

In order to improve the readability of a fuzzy rule base derived by a learning process pruning techniques can be used. Pruning techniques are well-known from neural networks and decision tree learning. They are used to reduce the complexity of a model.

Fuzzy rule base pruning can be based on a simple greedy algorithm that does not need to compute complex test values as, for example, in neural network pruning methods. In order to prune a rule base we consider four heuristic strategies that can work in an automatic fashion without the necessity of user interaction.

(i) Pruning by correlation: The variable that has the smallest influence on the output is deleted. To identify this variable statistical measures like correlations and χ^2 tests or measures from information theory like the information gain can be used.

(ii) Pruning by classification frequency: The rule that yields the largest degree of fulfilment in the least number of cases is deleted. Such a rule is only responsible for the classification of a small number of patterns. If these patterns are also covered by other rules, the performance of the fuzzy rule base may not decrease. But if these patterns represent exceptions it may not be possible to delete the selected rule without a decrease in performance.

(iii) Pruning by redundancy: The linguistic term that yields the minimal degree of membership in an active rule in the least number of cases is deleted. This pruning strategy assumes that the min operator is used in order to evaluate the antecedent of a rule. In this case a term that always provides large degrees of membership, does not influence the computation of the degree of fulfilment and the term assumes the role of a don't care variable. This pruning strategy can also be applied, if other t-norms are used, e.g. the product, but it may be less effective in these cases.

(iv) Pruning by fuzziness: The fuzzy set with the largest support is identified and all terms that use this fuzzy set are removed from the antecedents of all rules. This pruning strategy is comparable to (iii), because it assumes that fuzzy sets with large supports provide large degrees of membership for many input values and thus terms that use this fuzzy set do not influence the computation of the degree of fulfilment in a rule. Another justification for this strategy is that fuzzy sets actually get very large supports during training, if the corresponding variable has a large variance and is thus less useful for prediction.

An automatic pruning algorithm can be obtained by applying the four strategies consecutively. After each pruning step the membership functions should be trained again before a pruning step is declared a failure or a success. If a pruning step has failed, the rule base is restored to its previous state. This means the modifications caused by a pruning step are only kept, if the step has successfully improved the rule base. In the case of a classification problem, the pruning algorithm must take care not to remove the last rule for a class.

After an application step of one of the strategies has failed, it must be decided whether to carry on with this strategy and the next parameter it recommends for pruning, or to switch to the next pruning strategy. In order to reduce runtime, usually each of the four pruning strategies is iterated until a pruning step fails to improve the performance of the rule base. Then the next pruning strategy is selected. An implementation of this approach produces good results for neuro-fuzzy classification systems [25]. However, especially in high-dimensional problems exhaustive pruning can provide better results. In this case each pruning method is applied exhaustively and is not stopped when it fails for the first time. To speed up the process fuzzy set learning should be only done after an after exhaustive pruning process and not after each single pruning step. Exhaustive pruning can be iterated several time until the rule base cannot be reduced further.

The improvement of the rule base can be defined in terms of performance (i.e., reduction of error) and in terms of complexity or simplicity (i.e., number of parameters). There is usually a trade-off between performance and simplicity. To obtain high accuracy, a large number of free parameters is needed, which again result in a very complex and thus less comprehensible model. However, often the performance of a model can actually increase with the reduction of the number of parameters because the generalization capabilities of the model may increase. If the model has too many parameters, it tends to overfit the training data and displays poor generalization on test data. But if the number of parameters is too small, sufficient accuracy can no longer be attained.

If variables are deleted from the rule, the rule base can become inconsistent during pruning. This may happen for the above-mentioned pruning strategies (i), (iii) and (iv). Inconsistencies must be resolved by deleting some rules. If the rule learning algorithm shown in the previous section is applied the performance values of the rules can be used to select rules for deletion until the rule base is consistent again.

A consistent rule base is a rule base that does not contain contradictions or redundancies.

- A contradiction occurs if there are two rules with different consequents and their antecedents are either equal or one is more general than the other. An antecedent A is more general than an antecedent B, if A contains fewer linguistic terms than B and all linguistic terms of A also appear in B.
- A rule base is redundant, if there are two rules with identical consequents and if the antecedent of one rule is more general than the antecedent of the other rule.

The rule base can be made consistent by identifying pairs of contradictory and/or redundant rules and deleting rules with smaller performance values.

6. Application of NEFCLASS

In this section we demonstrate the NEFCLASS rule learning and pruning algorithms on two benchmark data sets: the "Wisconsin Breast Cancer" (WBC) and the "Pima Indian Diabetes" (PID) data sets. Both sets are available from the machine learning repository of the University of Irvine at ftp://ftp.ics.uci.edu/pub/machine-learning-databases. As the example for we use the "Wisconsin Breast Cancer" (WBC).

6.1. WBC data set

The WBC data set is a breast cancer database that was provided by Wolberg from the University of Wisconsin Hospitals, Madison [28]. The data set contains 699 cases and 16 of these cases have missing values. Each case is represented by an id number and nine attributes (x_1: clump thickness, x_2: uniformity of cell size, x_3: uniformity of cell shape, x_4: marginal adhesion, x_5: single epithelial cell size, x_6: bare nuclei, x_7: bland chromatin, x_8: normal nucleoli, x_9: mitoses). All attributes are from the domain $\{1, \ldots, 10\}$. Each case belongs to one of two classes (benign: 458 cases, or malignant: 241 cases).

The values of all nine variables are actually from an ordinal scale. Classifiers usually simply treat them as metric values and good classification results can be obtained this way (see Table 2). To illustrate Algorithm 1 we chose to interpret variables x_3 and x_6 as categorical variables and the rest as metric variables. x_3 and x_6 are selected, because these two variables usually turn out to be influential in other classification approaches.

We use a 10-fold cross validation, and let the tool select the best two rules per class during rule learning. For each metric variable two initial membership functions are given (shouldered triangles). The fuzzy sets for the categorical variables are created during rule learning. The fuzzy sets are trained until the error on the validation set reaches a local minimum which the algorithm cannot escape within 30 epochs (look-ahead). The maximum number of training cycles is set to 200 epochs and the learning rate is set to 0.1 (default value). We used exhaustive pruning and trained the fuzzy set only before and after pruning, but not after each pruning step.

The final classifier contains only two rules using one and two variables, respectively:

(i) if x_2 (uniformity of cell size) is *small* and x_6 (bare nuclei) is $term_1^{(6)}$ then *benign*

(ii) if x_2 (uniformity of cell size) is *large* then *malignant*

The membership functions after training are shown in Fig. 2. The fuzzy set for the categorical variable x_6 is drawn as a histogram. Its exact representation is

$$term_1^{(6)} = \{(1,1.0),(2,1.0),(3,0.66),(4,0.37),(5,0.61),(6,0.0),(7,0.01),(8,0.01),(9,0.0),(10,0.14)\}.$$

This classifier causes 28 misclassifications (4.01%) on the *training data*, i.e., its classification rate is 95.99% (see Table 1). This classifier covers all data, there are no unclassified cases. The error estimation for *unseen data* obtained from cross validation yields 4.58 ± 1.21% misclassifications, i.e., an estimated classification rate of 95.42 ± 1.21% (99% confidence interval). This error estimation must be interpreted this way: A classifier that is obtained by the described learning procedure and using the described parameters and training data is estimated to produce an error of 4.58 ± 1.21% on unseen data.

The final rule base was also discovered in one of the validation cycles. Altogether seven different rule bases were discovered during validation (nine rule bases with two rules, one rule base with four rules). However, most of the other

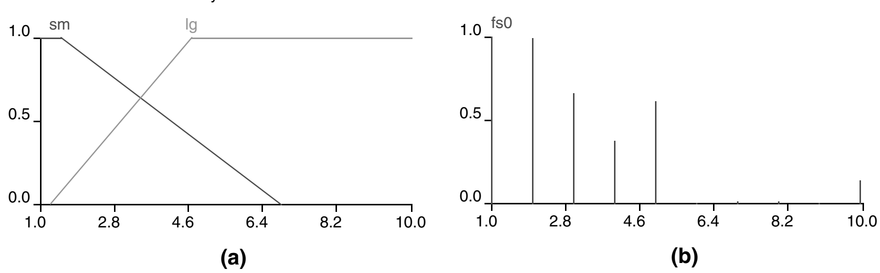

Fig. 2. Membership functions for the metric variable x_2 and the categorical variable x_6 after training.

Table 1
The confusion matrix of the final NEFCLASS-J classifier for the WBC data set

	Predicted class							
	Malignant		Benign		Not classified		Sum	
Malignant	228	(32.62%)	13	(1.86%)	0	(0.00%)	241	(34.99%)
Benign	15	(2.15%)	443	(63.38%)	0	(0.00%)	458	(65.01%)
Sum	243	(34.76%)	456	(65.24%)	0	(0.00%)	699	(100.00%)

Correct: 671 (95.99%), misclassified: 28 (4.01%), SSE: 60.54.

Table 2
Comparing the NEFCLASS learning outcome for the WBC data set to some other approaches

Model	Tool	Remarks	Error (%)	Validation
Discriminant analysis	SPSS	Linear model nine variables	3.95	1-Leave-out
Multilayer perceptron	SNNS	Four hidden units, RProp	5.18	50% Test set
Decision tree	C4.5	31 (24.4) Nodes, pruned	4.9	10-Fold
Rules from decision tree	C4.5 rules	8 (7.5) Rules using 1–3 Variables	4.6	10-Fold
NEFCLASS (metric variables)	NEFCLASS-J (Java version)	2 (2) Rules using 1–5 variables	5.86	10-Fold
NEFCLASS (two symbolic variables)	NEFCLASS-J (Java version)	2 (2.1) Rules using 1–3 variables	4.58	10-Fold

Numbers in ( ) are mean values from cross validation. The column "Error" contains an estimated error for unseen data.

rule bases were very similar and differed only in additionally using the other categorical variable x_3, using x_3 instead of x_2, or using just x_2.

Table 2 compares the result obtained with NEFCLASS-J (last entry) to results obtained with other approaches. The classification performance on unseen data is very good and the classifier is very compact. The error estimates given in the column "Validation" of Table 2 are either obtained from 1-leave-out cross validation (discriminant analysis), 10-fold cross validation, or from testing the solution once by withholding 50% of the data for a test set (MLP). Note that the cases with missing values had to be removed only for the MLP. All other approaches are able to handle missing values.

6.2. PID data set

The PID data set describes eight medical conditions for a selected group of 768 female Native Americans. There are two classes *positive* and *negative* describing the result of a test for diabetes. The majority class with 500 cases

(65.1%) is the *negative* class. For training NEFCLASS we used a learning rate of 1.0 and set 100 as the maximum number of epochs for fuzzy set learning with a look-ahead of 30 epochs. We used automatic best-per-class rule learning and exhaustive pruning. The fuzzy sets were not trained during pruning, only once before and after pruning. For each variable we used two fuzzy sets. During training a classifier was considered to have been improved if the number misclassifications could be reduced even if the mean square error has increased.

A 10-fold cross validation suggests that a NEFCLASS model will produce an error rate of 22.65 ± 3.53% on unseen data, i.e., the classification is expected to be correct in 77.35 ± 3.53%. The final pruned classifier was generated on the complete data set. It contains nine rules using between five and six variables and has a classification rate of 71.35% on the training data set. The confusion matrix of this classifier is given in Table 3.

To obtain the final classifiers we used the whole data set for training and obtained the following results.

- A multilayer perceptron with six hidden units trained for 300 epochs with resilient propagation results in a classification rate of 79.16% (160 errors). Because the learning outcome depends on the initial random weight configuration we picked the best network from 30 runs. Longer training did not yield better results.
- A pruned decision tree with 177 nodes using all nine attributes delivers a classification rate of 94.4% (43 errors).
- A linear discriminant function analysis uses five attributes to provide a classification rate of 77.3% (174 errors).
- NEFCLASS creates a classifier based on nine rules using between five and six variables and provides a classification rate of 71.35% (220 errors).

We can see that the other models can fit the complete training data set better. However, NEFCLASS provides the smallest estimated generalisation error during cross-validation. Only the linear discriminant function gives a similar result. This suggests that NEFCLASS – although it provides the worst performance on the training data – does generalise best on *unseen* data for this data set. This means, NEFCLASS would be a preferred classifier for this data

Table 3
The confusion matrix of the final NEFCLASS-J classifier generated on the complete PID data set

	Predicted class							
	Negative		Positive		Not classified		Sum	
Negative	437	(54.43%)	49	(9.51%)	14	(1.17%)	500	(65.10%)
Positive	142	(16.02%)	111	(17.06%)	15	(1.82%)	268	(34.90%)
Sum	579	(75.39%)	160	(20.83%)	29	(3.78%)	768	(100.00%)

Correct: 548 (71.35%), misclassified: 220 (28.65%), SSE: 338.57.

set, because it does not over-generalise on the training data compared to the expected accuracy on unseen data. This can be explained by the limited resources NEFCLASS is allowed to use for fitting the training data. In this example we only let NEFCLASS use two fuzzy sets per variable. In addition, NEFCLASS is not allowed to apply arbitrary modifications during training that could improve the fit. NEFCLASS is restricted by constraints that aim at generating meaningful fuzzy sets [20]. Neural nets and decision trees can usually fit a data set arbitrarily well, which means they are prone to over-generalisation. Although a linear discriminant function provides also a good performance in relation to the other approaches, NEFCLASS still offers the advantage of a relatively small interpretable fuzzy rule base of only nine rules. Further inspection of the learning outcome can also improve the result as we will show below.

Table 4 provides the details of the cross validation procedure for the four mentioned approaches. For the training of the neural network we used six hidden units and ran resilient propagation for 300 epochs. The decision trees we generated using information gain ratio as attribute selection measure and they were pruned with confidence level pruning. The discriminant analysis was run with a stepwise inclusion of variables using Wilk's lambda. For NEFCLASS we used the above-described settings.

By studying Table 4 we can see from the confidence interval of the error that the selection of training data seems to influence all approaches, especially the neural network. When we analyse the log file of the 10-fold cross validation for NEFCLASS we find four classifier with two or three rules and six classifiers with rule bases between five and eight rules. If we look for a small rule base with similar performance as the final classifier, we can identify a classifier with the following rule base:

- if body mass index is *large* and age is *large* then *positive*;
- if serum insulin is *large* and body mass index is *large* and diabetes pedigree function is *small* and age is *small* then *positive*;

Table 4
Comparing the NEFCLASS learning outcome for the PID data set to some other approaches

Model	Tool	Remarks	Error (%)	Validation
Discriminant analysis	SPSS	Linear model five variables	22.9	1-Leave-out
Multilayer perceptron	NNT[a]	Six hidden units, RProp	31.23 ± 7.2	10-Fold
Decision tree	DTI[a]	141–181 Nodes pruned	27.99 ± 4.7	10-Fold
NEFCLASS	NEFCLASS-J	2–8 Rules using 1–6 variables	22.65 ± 3.53	10-Fold

The column "Error" contains an estimated error and its 99% confidence interval for unseen data based on cross validation.

[a] Free software from Christian Borgelt, http://fuzzy.cs.uni-magdeburg.de/~borgelt.

- if serum insulin is *small* and body mass index is *small* and diabetes pedigree function is *small* then *negative*.

This rule base uses only four variables and has a classification rate of 71.6% (218 errors) on the whole data set. Apparently, the pruning algorithm of NEFCLASS is strongly influenced by the subsets used for training and validation. A similar pattern can be seen during cross validation for the decision trees, where we find trees that yield rule bases between 63 and 84 rules.

In situations like this it is an advantage to use a classifier like NEFCLASS that produces small and interpretable results. The user can review all solutions generated during cross validation and use domain knowledge to identify plausible solutions. The decision trees that are produced during cross validation are too large for manual inspection and the neural networks can hardly be interpreted at all.

If we select the above-given rule base of three rules and enter it as prior knowledge into NEFCLASS, we can obtain a classifier with a classification rate of 71.8% (216 errors) on the whole data set. We used only 50% of the data for training and reached a classification rate of 73.4% (102 errors) on the test set. The fuzzy sets for the four variables are shown in Fig. 3. This experiment shows that NEFCLASS is useful as an interactive tool for building small interpretable fuzzy classifiers. The user cannot necessarily expect to obtain the best solution automatically. But by inspecting several learning results and – if possible – combining them with domain knowledge, a user can obtain suitable solutions.

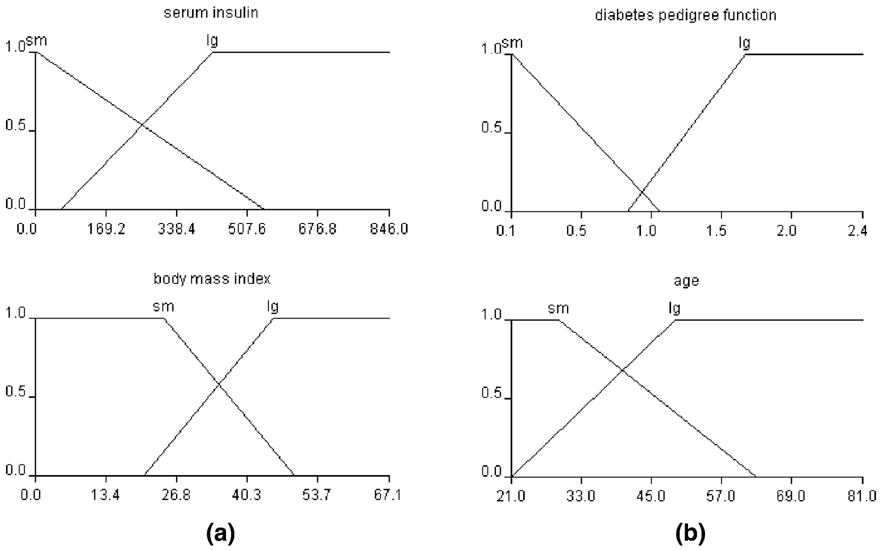

Fig. 3. Membership functions for four variables used in the experiment on the PID data set.

7. Conclusions

The interpretability of a fuzzy system – especially if applied in data analysis – is one of its key advantages. To support the readability of a fuzzy model resulting from a training process, one should use approaches that keep the learning algorithms simple – and therefore understandable – and do not touch the semantics of the underlying fuzzy models. The algorithms used by NEFCLASS conform to this idea.

We view neuro-fuzzy approaches as a way to heuristically find parameters of fuzzy models by processing training data with a learning algorithm. Neuro-fuzzy approaches should be seen as development tools that can help to construct a fuzzy model. They are not "automatic fuzzy model creators". The user should always supervise and interpret the learning process. This view matches the exploratory nature of IDA.

If we are only interested in using a neuro-fuzzy model for prediction, we could – from an applicational point of view – ask: why bother with interpretability and semantics? It is important that the model does its job. It is of course possible to leave out all constraints in the learning procedures of a neuro-fuzzy model, consider it only as a convenient tool that can be initialized by prior knowledge and trained by examples, and not look at the final model, as long as it performs to the satisfaction of the user. However, interpretability and clear semantics provide us with obvious advantages like checking the model for plausibility and maintaining it during its life cycle. These aspects are also important if a model is only to be used for prediction.

In order to applying a neuro-fuzzy learning strategy one more important aspect should be considered: for whatever reason we choose a fuzzy system to solve a problem it cannot be because we need an *optimal* solution. Fuzzy systems are used to exploit the tolerance for suboptimal solutions. So it does not make much sense to select a very sophisticated and expensive training procedure to squeeze the last bit of information from the training data. To do this we must usually forsake the standard fuzzy system architectures but, however, we are confronted with semantical problems instead. We prefer the view that fuzzy systems are used because they are easy to implement, easy to handle and easy to understand. A learning algorithm to create a fuzzy system from data should also have these features.

A free academic version of NEFCLASS is available at http://fuzzy.cs.uni-magdeburg.de/nefclass.

References

[1] B. Azvine, N. Azarmi, D. Nauck (Eds.), Intelligent Systems and Soft Computing: Prospects, Tools and Applications, Lecture Notes in Artificial Intelligence, vol. 1804, Springer, Berlin, 2000.

[2] H. Bersini, G. Bontempi, Now comes the time to defuzzify neuro-fuzzy models, Fuzzy Sets and Systems 90 (1997) 161–170.
[3] H. Bersini, G. Bontempi, M. Birattari, Is readability compatible with accuracy? From neuro-fuzzy to lazy learning, in: Fuzzy-Neuro Systems'98 – Computational Intelligence, Proceedings of the Fifth International Workshop on Fuzzy-Neuro-Systems'98 (FNS'98) in Munich, Germany, vol. 7 of Proceedings in Artifical Intelligence, infix, Sankt Augustin, 1998, pp. 10–25.
[4] M. Berthold, D.J. Hand (Eds.), Intelligent Data Analysis: An Introduction, Springer, Berlin, 1999.
[5] J. Casillas, O. Cordon, F. Herrera, L. Magdalena (Eds.), Trade-off between Accuracy and Interpretability in Fuzzy Rule-Based Modelling, Studies in Fuzziness and Soft Computing, Physica-Verlag, Heidelberg, 2002.
[6] O. Cordon, F. Herrera, I. Zwir, Analyzing and extending hierarchical systems of linguistic rules, in: Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, 2001, pp. 1121–1126.
[7] D.J. Hand, Intelligent data analysis: issues and opportunities, Int. J. Intelligent Data Analysis 2 (2) (1998), electronic journal. Available from <http://www.elsevier.com/locate/ida>.
[8] J.-S. Jang, C. Sun, E. Mizutani, Neuro Fuzzy and Soft Computing, Prentice Hall, Upper Saddle River, NJ, 1997.
[9] C.Z. Janikow, Fuzzy decision trees: issues and methods, IEEE Trans. Systems, Man Cybern. B 28 (1) (1998) 1–14.
[10] J.M. Keller, H. Tahani, Backpropagation neural networks for fuzzy logic, Inf. Sci. 62 (1992) 205–221.
[11] J. Kinzel, F. Klawonn, R. Kruse, Modifications of genetic algorithms for designing and optimizing fuzzy controllers, in: Proceedings of the IEEE Conference on Evolutionary Computation, IEEE Press, Orlando, FL, 1994, pp. 28–33.
[12] F. Klawonn, J. Gebhardt, R. Kruse, Fuzzy control on the basis of equality relations with an example from idle speed control, IEEE Trans. Fuzzy Systems 3 (3) (1995) 336–350.
[13] R. Kruse, J. Gebhardt, F. Klawonn, Foundations of Fuzzy Systems, Wiley, Chichester, 1994.
[14] R. Kruse, K.D. Meyer, Statistics with Vague Data, Reidel, Dordrecht, 1987.
[15] M. Lee, H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, in: Proceedings of the IEEE International Conference on Fuzzy Systems, San Francisco, March 1993, pp. 612–617.
[16] S. Mitra, L. Kuncheva, Improving classification performance using fuzzy mlp and two-level selective partitioning of the feature space, Fuzzy Sets and Systems 70 (1995) 1–13.
[17] D. Nauck, Using symbolic data in neuro-fuzzy classification, in: Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99), IEEE Press, New York, NY, 1999, pp. 536–540.
[18] D. Nauck, Adaptive rule weights in neuro-fuzzy systems, Neural Computing and Applications 9 (1) (2000a) 60–70.
[19] D. Nauck, Data analysis with neuro-fuzzy methods. Habilitation Thesis, Otto-von-Guericke University of Magdeburg, Faculty of Computer Science, Magdeburg, Germany, 2000b. Available from <http://www.neuro-fuzzy.de/∼nauck>.
[20] D. Nauck, F. Klawonn, R. Kruse, Foundations of Neuro-Fuzzy Systems, Wiley, Chichester, 1997.
[21] D. Nauck, R. Kruse, A neuro-fuzzy method to learn fuzzy classification rules from data, Fuzzy Sets and Systems 89 (1997) 277–288.
[22] D. Nauck, R. Kruse, NEFCLASS-X – a soft computing tool to build readable fuzzy classifiers, BT Technol. J. 16 (3) (1998) 180–190.
[23] D. Nauck, R. Kruse, Obtaining interpretable fuzzy classification rules from medical data, Artificial Intell. Med. 16 (2) (1999) 149–169.

[24] D. Nauck, R. Kruse, NEFCLASS-J – a java-based soft computing tool, in: B. Azvine, N. Azarmi, D. Nauck (Eds.), Intelligent Systems and Soft Computing: Prospects, Tools and Applications, Lecture Notes in Artificial Intelligence, vol. 1804, Springer, Berlin, 2000, pp. 143–164.
[25] D. Nauck, U. Nauck, R. Kruse, NEFCLASS for JAVA – new learning algorithms, in: Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99), IEEE Press, New York, 1999, pp. 472–476.
[26] S.K. Pal, S. Mitra, Multi-layer perceptron, fuzzy sets and classification, IEEE Trans. Neural Networks 3 (1992) 683–697.
[27] L.-X. Wang, J.M. Mendel, Generation rules by learning from examples, in: International Symposium on Intelligent Control, IEEE Press, New York, 1991, pp. 263–268.
[28] W. Wolberg, O. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, Proc. Natl. Acad. Sci. 87 (1990) 9193–9196.
[29] L.A. Zadeh, Fuzzy logic, neural networks and soft computing, Commun. ACM 37 (3) (1994) 77–84.
[30] L.A. Zadeh, Fuzzy logic and the calculi of fuzzy rules and fuzzy graphs: a precis, Int. J. Multiple-Valued Logic 1 (1996) 1–38.