ELSEVIER

Fuzzy Sets and Systems 89 (1997) 277–288

FUZZY sets and systems

A neuro-fuzzy method to learn fuzzy classification rules from data

Detlef Nauck*, Rudolf Kruse

Otto-von-Guericke-University of Magdeburg, Faculty of Computer Science, Universitätsplatz 2, D-39106 Magdeburg, Germany

Received 1 December 1996

Abstract

Neuro-fuzzy systems have recently gained a lot of interest in research and application. Neuro-fuzzy models as we understand them are fuzzy systems that use local learning strategies to learn fuzzy sets and fuzzy rules. Neuro-fuzzy techniques have been developed to support the development of e.g. fuzzy controllers and fuzzy classifiers. In this paper we discuss a learning method for fuzzy classification rules. The learning algorithm is a simple heuristics that is able to derive fuzzy rules from a set of training data very quickly, and tunes them by modifying parameters of membership functions. Our approach is based on NEFCLASS, a neuro-fuzzy model for pattern classification. We also discuss some results obtained by our software implementation of NEFCLASS, which is freely available on the Internet. © 1997 Elsevier Science B.V.

Keywords: Fuzzy classification; Neuro-fuzzy system

1. Introduction

The determination of fuzzy rules from data is an important issue for solving tasks like building fuzzy controllers, fuzzy classifiers, or supporting decision-making processes. Classification and decision support with the help of fuzzy rules is the background for this paper. In a lot of application areas like banking, insurance, medicine, etc. there are huge, sometimes unstructured data collections, and there is a need to transfer data into information. Currently, we observe new trends in data management like data warehousing and data mining. Several data bases of an organization are merged into a large data base, which represents several dimensions (views) of the data at the same time to support efficient data analysis without the need of computing complex joins. This so-called data warehouse also stores results of often needed calculations. On top of a data warehouse, data mining tools are used to analyze the data, and to transform it into information that supports decision making.

Several well-known methods are used in the process of data mining, like statistics, machine learning, neural networks and fuzzy data analysis. In this paper we consider the latter area, especially fuzzy classification, and we describe techniques for deriving fuzzy rules from data. The advantage of fuzzy rules for decision making lies in their interpretability. Decision makers in industry are usually not statisticians, mathematicians, or AI experts. So it is important, that the results of the data analysis process is presented in a form that can be easily grasped by non-experts. The linguistic representation of fuzzy rules supports this approach.

For learning classification rules from data we use our NEFCLASS (neuro-fuzzy classification) model

* Corresponding author. Tel.: +49.391.67.12700, fax: +49.391.67.12018; e-mail: nauck@iik.cs.uni-magdeburg.de; URL: fuzzy.cs.uni-magdeburg.de/~nauck.

PII S0165-0114(97)00009-2

[20] for data analysis problems. NEFCLASS is used to derive fuzzy classification rules from a set of data that can be separated in different crisp classes. The fuzzy rules describing the data are of the form:

if x_1 is μ_1 **and** x_2 is μ_2 **and** ... **and** x_n is μ_n
then pattern $(x_1, x_2, \ldots, x_n)$ belongs to class C,

where $\mu_1, \ldots, \mu_n$ are fuzzy sets. The task of the NEFCLASS model is to discover these rules, and to learn the shape of the membership functions. The patterns are vectors $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and a class C is a (crisp) subset of $\mathbb{R}^n$. We assume the intersection of two different classes to be empty. The pattern feature values are represented by fuzzy sets, and the classification is described by a set of linguistic rules. For each input feature x_i there are q_i fuzzy sets $\mu_1^{(i)}, \ldots, \mu_{q_i}^{(i)}$, and the rule base contains k fuzzy rules $R_1, \ldots, R_k$. NEFCLASS does not induce fuzzy classification rules by searching for clusters, but by modifying the fuzzy partitionings defined on each single dimension. Other approaches for the generation of fuzzy rules from data usually search for hyper-ellipsoidal [13] or hyper-rectangular clusters [1, 2, 24]. Fuzzy sets are then constructed by projecting the clusters.

Fuzzy clustering methods search for hyper-spheres (FCM) or hyper-ellipsoidal clusters in the pattern space [4, 5]. A single pattern may belong to several clusters at once to each with a different degree of membership. The membership values of one pattern over all clusters must usually add up to 1 (probabilistic interpretation), but in some approaches this property is not enforced (possibilistic interpretation). Each cluster represents a fuzzy rule. To obtain an interpretable linguistic rule, it is necessary to construct one-dimensional fuzzy sets for each input feature. They are created by projecting the clusters to each single dimension. However, this method tends to produce rules which are hard to interpret, because fuzzy sets are produced individually for each rule. It is therefore difficult to label the fuzzy sets with linguistic terms. In the case of hyper-ellipsoidal clusters the projection also causes a loss of information [14]. This can only be completely avoided, when hyper-rectangular clusters are used.

Models that search for hyper-rectangular clusters (hyper-boxes) tend to produce a lot of clusters, but usually classify a pattern set correctly after only a few runs through the training data. A new hyper-box is created each time a misclassified pattern is found. If the hyper-boxes must not overlap, because the model must produce crisp rules (i.e. RuleNet [25]), neighboring hyper-boxes must be corrected accordingly. If fuzzy rules are sought, then the hyper-boxes may overlap (e.g. Fuzzy RuleNet [24]), but the degree of overlapping must be watched. Rules are created by projecting the hyper-boxes on each single dimension, thus creating fuzzy sets. This method does not cause a loss in information. However, like in fuzzy clustering, fuzzy sets are created individually for each rule. The result may be a rule base which is hard to interpret.

If performance in classification is not the only issue, but interpretation of the classification result is also important, than another method of fuzzy rule creation is of interest. NEFCLASS ensures that a linguistic term is modeled by only one fuzzy set, i.e. fuzzy rules are created out of given initial fuzzy sets, and not vice versa. NEFCLASS does not search for clusters in the input space, but covers it with a grid created by predefined fuzzy partitionings over each single dimension. The learning algorithm modifies the fuzzy sets directly, and can therefore use constraints to make sure that the induced fuzzy rules can be interpreted well, and that the fuzzy sets can be more easily labeled with linguistic expressions. However, the rule generation capabilities of our neuro-fuzzy approach is more restricted than fuzzy clustering approaches. For some problems it may be difficult to obtain a sufficient initial rule base by its learning algorithm alone. If prior rule-based knowledge is not available to initialize the model, it can be useful to obtain an initial rule base by fuzzy clustering methods, which is then completed and refined by the neuro-fuzzy learning algorithm.

In the following sections we will review our neuro-fuzzy approach NEFCLASS [20] and present its learning algorithms for the creation of fuzzy rules, and the adaption of membership functions. Then we discuss some learning results, and show how to initialize a NEFCLASS system with rules from fuzzy clustering.

2. A fuzzy classifier in a neuro-fuzzy architecture

We want to create a fuzzy classifier from data by learning techniques. We will therefore use a neuro-fuzzy architecture to represent the classifier. This

provides us with a framework that makes the application of local learning procedures more obvious, and helps to compare different approaches in this area.

Modern neuro-fuzzy-systems are usually represented as a multilayer feedforward neural network [3, 7, 8, 11, 21, 22, 19, 24], but fuzzifications of other neural network architectures are also considered, for example, self-organizing feature maps [6, 26]. In neuro-fuzzy models, connection weights and propagation and activation functions differ from common neural networks. Although there are a lot of different approaches [7, 22], we only want to consider neuro-fuzzy systems which display the following properties:

(i) A neuro-fuzzy system is based on a fuzzy system which is trained by a learning algorithm derived from neural network theory. The (heuristical) learning procedure operates on local information, and causes only local modifications in the underlying fuzzy system.

(ii) A neuro-fuzzy system can be viewed as a 3-layer feedforward neural network. The first layer represents input variables, the middle (hidden) layer represents fuzzy rules and the third layer represents output variables. Fuzzy sets are encoded as (fuzzy) connection weights. (*Remark*: Sometimes a 5-layer architecture is used, where the fuzzy sets are represented in the units of the second and fourth layer).

(iii) A neuro-fuzzy system can be always (i.e. before, during and after learning) interpreted as a system of fuzzy rules. It is also possible to create the system out of training data from scratch, as it is possible to initialize it by prior knowledge in form of fuzzy rules. (*Remark*: Not all neuro-fuzzy models support fuzzy rule creation).

(iv) 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 applicable to the system parameters. (*Remark*: Not all neuro-fuzzy approaches have this property).

(v) A neuro-fuzzy system approximates an n-dimensional (unknown) function that is partially defined by the training data. The fuzzy rules encoded within the system represent vague samples, and can be viewed as 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 [15].

In [21] we present a generic model for neuro-fuzzy systems based on the idea of a generic 3-layer fuzzy perceptron. The term "fuzzy (multi-layer) perceptron" has also been used by other authors for their approaches [12, 23, 16]. We use our interpretation of this notion here to describe the structure of our generic model. Other definitions of the term "fuzzy perceptron" are possible.

By using a generic fuzzy perceptron to derive neuro-fuzzy systems for special domains, it would be possible to evaluate these different neuro-fuzzy approaches by means of the same underlying model. The fuzzy perceptron was used to derive the NEFCON model [17] for neuro-fuzzy control applications, and also to define the NEFCLASS model [20] discussed in this paper. A generic fuzzy perceptron has the architecture of a usual multilayer perceptron, but 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 intention of this model is to provide a framework for learning algorithms, to be interpretable in form of linguistic rules, and to be able to use prior rule-based knowledge, so the learning has not to start from scratch.

Definition 1. A generic *3-layer fuzzy perceptron* is a 3-layer feedforward neural network $(U, W, \mathrm{NET}, A, O, \mathrm{ex})$ with the following specifications:

(i) $U = \bigcup_{i \in M} U_i$ is a non-empty set of units (neurons) and $M = \{1, 2, 3\}$ is the index set of U. For all $i, j \in M$, $U_i \neq \emptyset$, and $U_i \cap U_j = \emptyset$ with $i \neq j$ holds. U_1 is called input layer, U_2 rule layer (hidden layer), and U_3 output layer.

(ii) The structure of the network (connections) is defined as $W : U \times U \to \mathscr{F}(\mathbb{R})$, such that there are only connections $W(u, v)$ with $u \in U_i$, $v \in U_{i+1}$ $(i \in \{1, 2\})$ ($\mathscr{F}(\mathbb{R})$ is the set of all fuzzy subsets of $\mathbb{R}$).

(iii) A defines an activation function A_u for each $u \in U$ to calculate the activation a_u

(a) for input and rule units $u \in U_1 \cup U_2$:

$$A_u : \mathbb{R} \to \mathbb{R}, \quad a_u = A_u(\mathrm{net}_u) = \mathrm{net}_u,$$

(b) for output units $u \in U_3$:

$$A_u : \mathscr{F}(\mathbb{R}) \to \mathscr{F}(\mathbb{R}),$$
$$a_u = A_u(\mathrm{net}_u) = \mathrm{net}_u.$$

(iv) O defines for each $u \in U$ an output function O_u to calculate the output o_u

(a) for input and rule units $u \in U_1 \cup U_2$:

$$O_u : \mathbb{R} \to \mathbb{R}, \quad o_u = O_u(a_u) = a_u,$$

(b) for output units $u \in U_3$:

$$O_u : \mathscr{F}(\mathbb{R}) \to \mathbb{R},$$

$$o_u = O_u(a_u) = \mathrm{DEFUZZ}_u(a_u),$$

where DEFUZZ_u is a suitable defuzzification function.

(v) NET defines for each unit $u \in U$ a propagation function NET_u to calculate the net input net_u

(a) for input units $u \in U_1$:

$$\mathrm{NET}_u : \mathbb{R} \to \mathbb{R}, \quad \mathrm{net}_u = \mathrm{ex}_u,$$

(b) for rule units $u \in U_2$:

$$\mathrm{NET}_u : (\mathbb{R} \times \mathscr{F}(\mathbb{R}))^{U_1} \to [0,1],$$

$$\mathrm{net}_u = \top_{u' \in U_1} \{W(u',u)(o_{u'})\},$$

where $\top$ is a t-norm,

(c) for output units $u \in U_3$:

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

$$\mathrm{net}_u : \mathbb{R} \to [0,1],$$

$$\mathrm{net}_u(x) = \bot_{u' \in U_2} \{\top(o_{u'}, W(u',u)(x))\},$$

where $\bot$ is a t-conorm.

If the fuzzy sets $W(u',u)$, $u' \in U_2$, $u \in U_3$, are monotonic on their support, and $W^{-1}(u',u)(\tau) = x \in \mathbb{R}$ such that $W(u',u)(x) = \tau$ holds, then the propagation function net_u of an output unit $u \in U_3$ can alternatively be defined as

$$\mathrm{net}_u(x) = \begin{cases} 1 & \text{if } x = \dfrac{\sum_{u' \in U_2} o_{u'} \cdot m(o_{u'})}{\sum_{u' \in U_2} o_{u'}} \\ 0 & \text{otherwise} \end{cases}$$

with $m(o_{u'}) = W^{-1}(u',u)(o_{u'})$. To calculate the output o_u in this case

$$o_u = x, \text{ with } \mathrm{net}_u(x) = 1.$$

is used instead of (iv(b)).

(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.

A fuzzy perceptron can be viewed as a usual 3-layer perceptron that is "fuzzified to a certain extent". Only the weights, the net inputs, and the activations of the output units are modeled as fuzzy sets. A fuzzy perceptron is like a usual perceptron used for function approximation. The advantage lies within the interpretation of its structure in the form of linguistic rules, because the fuzzy weights can be associated with linguistic terms. The network can also be created partly, or in the whole, out of linguistic (fuzzy if–then) rules.

A NEFCLASS system is a special neuro-fuzzy architecture for classification problems. The following definition shows how it is derived from a generic 3-layer fuzzy perceptron.

Definition 2. A NEFCLASS system is a 3-layer fuzzy perceptron with the following specifications:

(i) $U_1 = \{x_1, \ldots, x_n\}$, $U_2 = \{R_1, \ldots, R_k\}$, and $U_3 = \{c_1, \ldots, c_m\}$.

(ii) Each connection between units $x_i \in U_1$ and $R_r \in U_2$ is labeled with a linguistic term $A_{j_r}^{(i)}$ $(j_r \in \{1, \ldots, q_i\})$.

(iii) $W(R,c) \in \{0,1\}$ holds for all, $R \in U_2$, $c \in U_3$.

(iv) Connections coming from the same input unit x_i and having identical labels, bear the same weight at all times. These connections are called *linked connections*, and their weight is called *shared weight*.

(v) Let $L_{x,R}$ denote the label of the connection between the units $x \in U_1$ and $R \in U_2$. For all $R, R' \in U_2$ holds:

$$(\forall (x \in U_1)\ L_{x,R} = L_{x,R'}) \Rightarrow R = R'.$$

(vi) For all rule units $R \in U_2$ and all units $c, c' \in U_3$ we have

$$(W(R,c) = 1) \wedge (W(R,c') = 1) \Rightarrow c = c'.$$

(vii) For all output units $c \in U_3$, $o_c = a_c = \mathrm{net}_c$ holds.

(viii) For all output units $c \in U_3$ the net input net_c is calculated by

$$\mathrm{net}_c = \frac{\sum_{R \in U_2} W(R,c) \cdot o_R}{\sum_{R \in U_2} W(R,c)}.$$

The first layer U_1 of a NEFCLASS system contains the input units representing the pattern features. The activation a_x of a unit $x \in U_1$ is usually equal to its

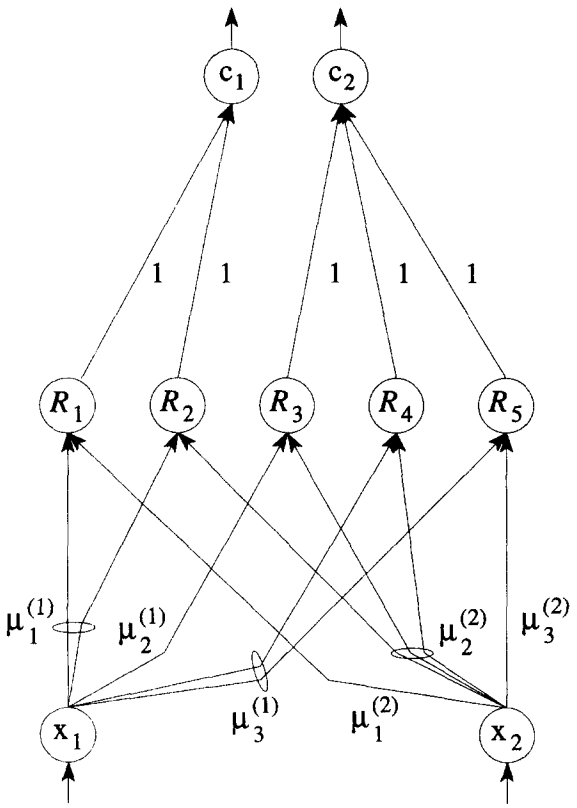

Fig. 1. A NEFCLASS system with two inputs, five rules and two output classes.

external input. However, it can be different, if the input unit does some kind of preprocessing (normalization, etc.). The hidden layer U_2 holds rule units representing the fuzzy rules, and the third layer U_3 consists of output units, one for each class. The weights on the connections from rule units to output units are fixed at 1 for semantical reasons [20]. Alternatively, the network input to an output unit can be computed by a maximum operation instead of a weighted sum. This can be selected in accordance to the application problem. Instead of min and max, other t-norms and t-conorms can be used.

The rule base is an approximation of an (unknown) function $\varphi : \mathbb{R}^n \to \{0,1\}^m$ that represents the classification task where $\varphi(\boldsymbol{x}) = (c_1, \ldots, c_m)$ such that $c_i = 1$ and $c_j = 0$ $(j \in \{1, \ldots, m\},\ j \neq i)$, i.e. $\boldsymbol{x}$ belongs to class C_i. Because of the mathematics involved the rule base actually does not approximate φ but the function $\varphi' : \mathbb{R}^n \to [0,1]^m$. We will get $\varphi(\boldsymbol{x})$ by $\varphi(\boldsymbol{x}) = \psi(\varphi'(\boldsymbol{x}))$, where ψ reflects the interpretation of the classification result obtained from a NEFCLASS system. In our case we will map the highest component of each vector $\boldsymbol{c}$ to 1 and its other components to 0 (winner takes all).

The fuzzy sets and the linguistic rules which perform this approximation, and define the resulting NEFCLASS system, will be obtained from a set of examples by learning. Fig. 1 shows a NEFCLASS system that classifies input patterns with two features into two distinct classes by using five linguistic rules.

Its main feature are the *shared weights* on some of the connections. This way we make sure, that for each linguistic value (e.g. "x_1 is *positive big*") there is only one representation as a fuzzy set (e.g. $\mu_1^{(1)}$ in Fig. 1), i.e. the linguistic value has only one interpretation for all rule units (e.g. R_1 and R_2 in Fig. 1). It cannot happen that two fuzzy sets that are identical at the beginning of the learning process develop differently, and so the semantics of the rule base encoded in the structure of the network is not affected [19]. Connections that share a weight always come from the same input unit.

3. Learning fuzzy rules and fuzzy sets

A NEFCLASS system can be build from partial knowledge about the patterns, and can be then refined by learning, or it can be created from scratch by learning. A user has to define a number of initial fuzzy sets partitioning the domains of the input features, and must specify a value for k, i.e. the maximum number of rule nodes that may be created in the hidden layer.

The idea of the learning algorithm is to first create a rule base, and then refining it by modifying the initially given membership functions (usually fuzzy partitions where the membership degrees of each value add up to 1.0). The rule base will be created by finding for each pattern in the training set a rule that best classifies it. If a rule with an identical antecedent is not already in the rule base, it will be added. Definition 3 presents three such strategies to find a rule base. The learning algorithm of the membership functions uses an error measure that tells, whether the degree of fulfillment of a rule has to be higher or lower. This information is used to change the input fuzzy sets.

The learning algorithm for NEFCLASS is described in the following definition. We assume that triangular membership functions are used that are described by three parameters:

$$\mu : \mathbb{R} \to [0,1], \quad \mu(x) = \begin{cases} \dfrac{x-a}{b-a} & \text{if } x \in [a,b), \\ \dfrac{c-x}{c-b} & \text{if } x \in [b,c], \\ 0 & \text{otherwise.} \end{cases}$$

In addition, the leftmost and rightmost membership functions for each variable can be shouldered.

Definition 3 (*NEFCLASS learning algorithm*). Consider a NEFCLASS system with n input units $x_1,\dots,x_n$, $k \leqslant k_{\max}$ rule units $R_1,\dots,R_k$, and m output units $c_1,\dots,c_m$. Also given is a learning set $\tilde{\mathcal{L}} = \{(\boldsymbol{p}_1,\boldsymbol{t}_1),\dots,(\boldsymbol{p}_s,\boldsymbol{t}_s)\}$ of s patterns, each consisting of an input pattern $\boldsymbol{p} \in \mathbb{R}^n$, and a target pattern $\boldsymbol{t} \in \{0,1\}^m$. The learning algorithm that is used to create the k rule units of the NEFCLASS system consists of the following steps
(*rule learning algorithm*):

(i) Select the next pattern $(\boldsymbol{p},\boldsymbol{t})$ from $\tilde{\mathcal{L}}$

(ii) For each input unit $x_i \in U_1$ find the membership function $\mu_{j_i}^{(i)}$ such that

$$\mu_{j_i}^{(i)}(p_i) = \max_{j \in \{1,\dots,q_i\}} \{\mu_j^{(i)}(p_i)\}.$$

(iii) If there are still less than $k_{\max}$ rule nodes, and there is no rule node R with

$$W(x_1,R) = \mu_{j_1}^{(1)},\dots,W(x_n,R) = \mu_{j_n}^{(n)}$$

then create such a node, and connect it to the output node c_l if $t_l = 1$.

(iv) If there are still unprocessed patterns in $\tilde{\mathcal{L}}$, and $k < k_{\max}$ then proceed with step (i), and stop otherwise.

(v) Determine the rule base by one of the following three procedures:

- "*Simple*" *rule learning*: Keep just the first k created rules (i.e. stop rule creation when $k_{\max} = k$ rules have been created).
- "*Best*" *rule learning*: Process $\tilde{\mathcal{L}}$ again, and accumulate the activations of each rule unit for each class of the propagated patterns. If a rule unit R displays a higher accumulated activation for a class C_j than for the class C_R specified by the rule conclusion, then change the conclusion of R to C_j, i.e. connect R to output unit c_j. Process $\tilde{\mathcal{L}}$ again, and compute for each rule unit

$$V_R = \sum_{p \in \tilde{\mathcal{L}}} a_R^{(p)} \cdot e_p,$$

$$e_p = \begin{cases} 1, & \text{if } p \text{ is classified correctly,} \\ -1, & \text{otherwise.} \end{cases}$$

Keep those k rule units with the highest values for V_R, and delete the other rule units from the NEFCLASS system.
- "*Best per class*" *rule learning*: Proceed like in "Best" rule learning, but keep for each class C_j those $\lfloor k/m \rfloor$ best rules whose conclusions represent the class C_j ($\lfloor x \rfloor$ is the integer part of x).

The supervised learning algorithm of a NEFCLASS system to adapt its fuzzy sets runs cyclically through the learning set $\tilde{\mathcal{L}}$ by repeating the following steps until a given end criterion is met (*fuzzy set learning algorithm*):

(i) Select the next pattern $(\boldsymbol{p},\boldsymbol{t})$ from $\tilde{\mathcal{L}}$, propagate it through the NEFCLASS system, and determine the output vector $\boldsymbol{c}$.

(ii) For each output unit c_i: Determine the delta value $\delta_{c_i} = t_i - a_{c_i}$.

(iii) For each rule unit R with $a_R > 0$:

(a) Determine the delta value

$$\delta_R = a_R(1 - a_R) \sum_{c \in U_3} W(R,c)\delta_c.$$

(b) Find x' such that

$$W(x',R)(a_{x'}) = \min_{x \in U_1} \{W(x,R)(a_x)\}.$$

(c) For the fuzzy set $W(x',R)$ determine the delta values for its parameters a,b,c using the learning rate $\sigma > 0$:

$$\delta_b = \sigma \cdot \delta_R \cdot (c-a) \cdot \operatorname{sgn}(a_{x'} - b),$$

$$\delta_a = -\sigma \cdot \delta_R \cdot (c-a) + \delta_b,$$

$$\delta_c = \sigma \cdot \delta_R \cdot (c-a) + \delta_b,$$

and apply the changes to $W(x', R)$ if this does not violate against a given set of constraints Φ. (Note: the weight $W(x', R)$ might be shared by other connections, and in this case might be changed more than once).

(iv) If an epoch was completed, and the end criterion is met, then stop; otherwise proceed with step (i).

There are three ways to create a rule base for a NEFCLASS system. The "simple" procedure can only be successful, if the patterns are selected randomly from the learning set, and if the cardinalities of the classes are approximately equal. It works for simple problems like the Iris data. Usually, a user will choose "best" or "best per class" rule learning. The latter procedure should be selected, when one supposes that the patterns are distributed in an equal number of clusters per class. "Best" rule learning is suitable, when there are classes, which have to be represented by a larger number of rules than other classes. Either way, rule learning is completed after three cycles through the data set.

The learning procedure for the fuzzy sets is a simple heuristics. It results in shifting the membership functions, and in making their supports larger or smaller (see Fig. 2). By changing only the fuzzy set that delivered the smallest membership degree for the current pattern, the changes are kept as small as possible. It is easy to define constraints Φ for the learning procedure, e.g. that fuzzy sets must not pass each other, or that they must intersect at 0.5, etc. Constraints like that help to obtain an interpretable rule base, but may cause a loss of performance in classification.

The sum in step (iii(a)) is not really necessary, because each rule unit is connected to only one output unit. But it makes the model more flexible, because it is possible to also use adaptive rule weights. We refrain from it, because we want to keep the semantics of a NEFCLASS system [20], and it is not clear what a weighted fuzzy rule is supposed to mean. Rule weights are often superfluous, because they can be represented as changes in the membership functions. We also found that rule weights are not necessary to obtain good classification results. However, without rule weights a NEFCLASS system usually cannot produce exact output values of 0 or 1 due to the mathematics involved. For the same reason the learning procedure cannot reach an error value of zero, and therefore the change in error is used as a stop criterion for the learning algorithm.

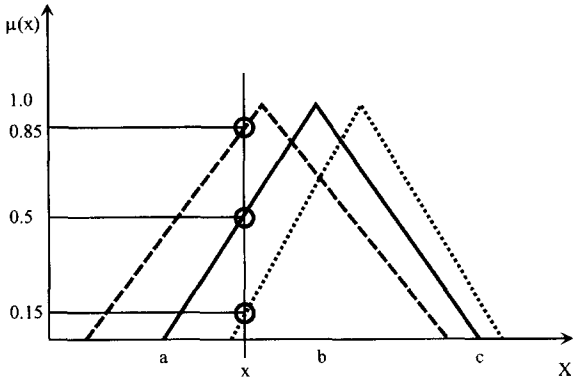

Fig. 2. The adaptation of fuzzy sets is carried out by simply changing the parameters of its membership function in a way that the membership degree for the current feature value is increased or decreased (middle: initial situation, left: increase situation, right: decrease situation).

The learning process is visualized in Fig. 3. The left part shows the situation after the rule learning algorithm has terminated (let us assume "best" rule learning, and $k_{\max} = 3$). The predefined fuzzy partitioning on both input variables defines a grid in the input space. This grid is created by overlapping hyper-boxes, where each hyper-box is formed by the Cartesian product of the supports of n fuzzy sets. Each hyper-box H_j represents the support of an n-dimensional fuzzy set, i.e. the antecedent of a fuzzy rule R_j. During rule learning a hyper-box H_j is selected, if at least one pattern has its highest membership degree with R_j. Each hyper-box is mapped to the class of the pattern which caused its selection (i.e. the rule conclusion is determined). After all patterns are processed, the mapping of hyper-boxes to classes is re-evaluated, and changed, where necessary. After this the $k_{\max}$ "best" hyper-boxes (fuzzy rules) are kept.

After rule learning there are usually some patterns which are not classified, because their hyper-box (rule) was not included in the set of $k_{\max}$ best rules. There are usually also some misclassifications. It is the task of the fuzzy set learning algorithm to improve this situation. By modifying the membership functions, the predefined grid is distorted. This results in the situation shown in the right part of Fig. 3.

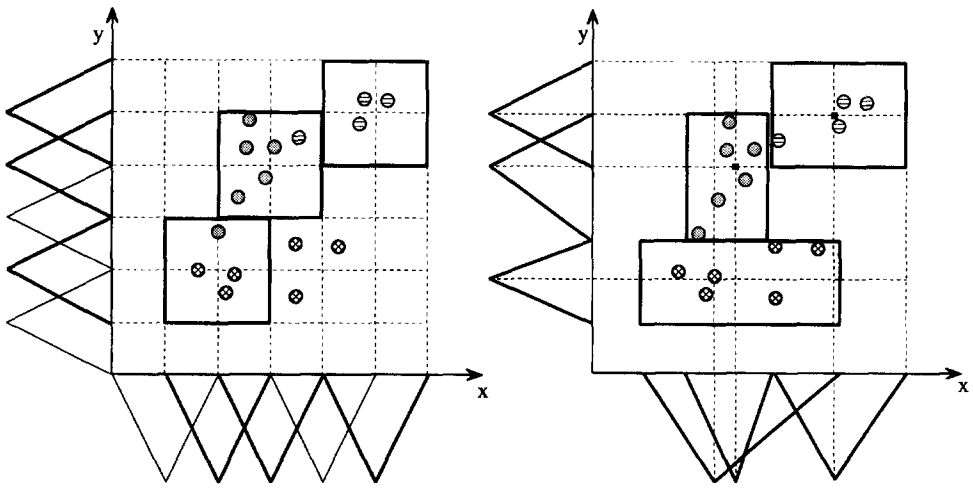

Fig. 3. Visualization of a possible NEFCLASS learning process after rule creation (left), and after fuzzy set tuning (right).

Because the learning algorithm for the fuzzy sets is constrained (e.g. a fuzzy set must not pass a neighbor), it is possible, that some changes to the form of the hyper-boxes are not applicable. This is one reason that there can be some remaining classification errors. Another reason can be a too small number of fuzzy rules. This can also lead to undesired forms of membership functions (e.g. too much overlapping). Considering the resulting situation in the right part of Fig. 3, in the given example, it is probably better to accept four instead of three rules to avoid the extremely wide support of the leftmost fuzzy set over feature x.

Compared to neural networks, NEFCLASS uses a much simpler learning strategy. There is no vector quantization involved to find rules (clusters), and the membership functions are not trained by gradient descent. Fuzzy rule creation can be seen as a selection from an initially given rule base, specified by a fuzzy grid in the input domain.

From the viewpoint of the NEFCLASS architecture and the flow of data, the fuzzy sets are trained by a backpropagation-like algorithm: the error is propagated from the output units towards the input units, and is used to change the membership function parameters, but there is no gradient information involved. The adaptivity of a NEFCLASS system is restricted, because of the initially given input fuzzy partitions, which define the form and maximal number of clusters, and by the constraints that do not admit certain changes in the fuzzy sets.

4. Deriving fuzzy rules from data

In this section we will present some results of a NEFCLASS system applied to classification problems. We will also show how NEFCLASS can benefit from initialization by fuzzy clustering. The first task is to obtain classification rules to correctly classify the patterns of the well-known Iris data set [9]. This data set contains 150 patterns belonging to three different classes (Iris Setosa, Iris Versicolour, and Iris Virginica) with 50 patterns each. The patterns have four input features (sepal and petal length and width of the iris flower). The first class is linearly separable from the other two classes, whereas the second and third class are not linearly separable from each other. This is a very simple problem, but it is often used as a benchmark.

For the NEFCLASS learning process the data set was split in half, and the patterns were ordered alternately within the training and test data sets. At first, we allowed the system to create a maximum of seven rules using the "best" rule learning algorithm. The domains of the four input variables were initially each partitioned by three equally distributed fuzzy sets. NEFCLASS created at first 19 rules (81 would be

Table 1
Seven rules found by NEFCLASS to classify the Iris data

if x_1 is *sm* and x_2 is *md* and x_3 is *sm* and x_4 is *sm*, **then** *Setosa*
if x_1 is *sm* and x_2 is *sm* and x_3 is *sm* and x_4 is *sm*, **then** *Setosa*
if x_1 is *md* and x_2 is *sm* and x_3 is *lg* and x_4 is *lg*, **then** *Virginica*
if x_1 is *lg* and x_2 is *sm* and x_3 is *lg* and x_4 is *lg*, **then** *Virginica*
if x_1 is *md* and x_2 is *sm* and x_3 is *md* and x_4 is *md*, **then** *Versicolour*
if x_1 is *lg* and x_2 is *md* and x_3 is *lg* and x_4 is *lg*, **then** *Virginica*
if x_1 is *md* and x_2 is *sm* and x_3 is *md* and x_4 is *sm*, **then** *Versicolour*

possible) and selected the best 7 rules (see Table 1). Fuzzy set learning stopped after 126 epochs, because the error was not decreased for 50 epochs. After learning, 3 out of 75 patterns from the training set were still classified wrongly (i.e. 96% correct). Testing the second data set, the NEFCLASS system classified only 2 out of 75 patterns incorrectly (i.e. 97.3% correct). Considering all 150 patterns the system performed well with 96.67% correct classifications.

Because the Iris problem is extremely simple, it can also be solved by less rules using for example only two of the four input variables. By looking at the two-dimensional plots of the input data, it can be easily seen that the third and fourth variable are sufficient to classify the data. A correlation analysis also shows that feature x_3 and x_4 have the highest correlations with the class information. We trained another NEFCLASS system, using only the third and fourth input, and allowed the system to create three rules. Using "best per class" rule learning, the system finds at first five rules (nine would be possible) and finally selects the three rules shown in Table 2. After training for 110 epochs, we get two errors on the training set, and three errors on the test set, i.e. the same performance like seven rules using all four features. The learning result is also shown in Fig. 4.

Due to its simple nature the Iris data can also be classified by using only one input feature and three fuzzy sets. This can be easily seen in the two-dimensional data plot in Fig. 4. When we use only feature x_4, which has the highest correlation to the class information (0.96), we can train NEFCLASS to produce three (of three possible) rules by using either rule learning procedure. In this case we get $6(2+4)$ classification errors on the whole data set.

We have compared the learning result of NEFCLASS to the results obtained with the FuNe-I system presented in [11]. This neuro-fuzzy system reached a classification rate of 99% on the test set of the Iris data using 13 rules and four inputs, and a 96% classification rate using 7 rules and 3 inputs. FuNe I is offered in a limited test version by the authors of [11], so we could run our own test to compare it with NEFCLASS. We allowed the system to create 10 rules, and it came out with 5 classification errors on the test set after training. Therefore, the two models are comparable in their performance, even though FuNe I has a much more complex structure and training procedure. FuNe I also uses a concept of weighted rules. We refrained from using this approach in NEFCLASS, because the semantics of weighted fuzzy rules is not clear [18].

Table 2
Five classification rules found by NEFCLASS when only the last two features of the Iris data are used. The first three rules were selected by the rule learning procedure "best per class"

No.	Classification rule	Scoring
1	**if** x_3 is *sm* and x_4 is *sm*, **then** *Setosa*	23.80
2	**if** x_3 is *md* and x_4 is *md*, **then** *Versicolour*	7.47
3	**if** x_3 is *lg* and x_4 is *lg*, **then** *Virginica*	15.53
4	**if** x_3 is *lg* and x_4 is *md*, **then** *Virginica*	0.13
5	**if** x_3 is *md* and x_4 is *sm*, **then** *Versicolour*	4.53

As a second demonstration of the NEFCLASS learning algorithms we use the "Wisconsin Breast Cancer" data set [27]. This data set contains 699 cases distributed into two classes (benign and malign). We used only 683 cases, because 16 cases have missing values. Each pattern has nine features. To show how NEFCLASS performs when prior knowledge is supplied, we used a fuzzy clustering method to obtain fuzzy rules. We used a modification of the algorithm

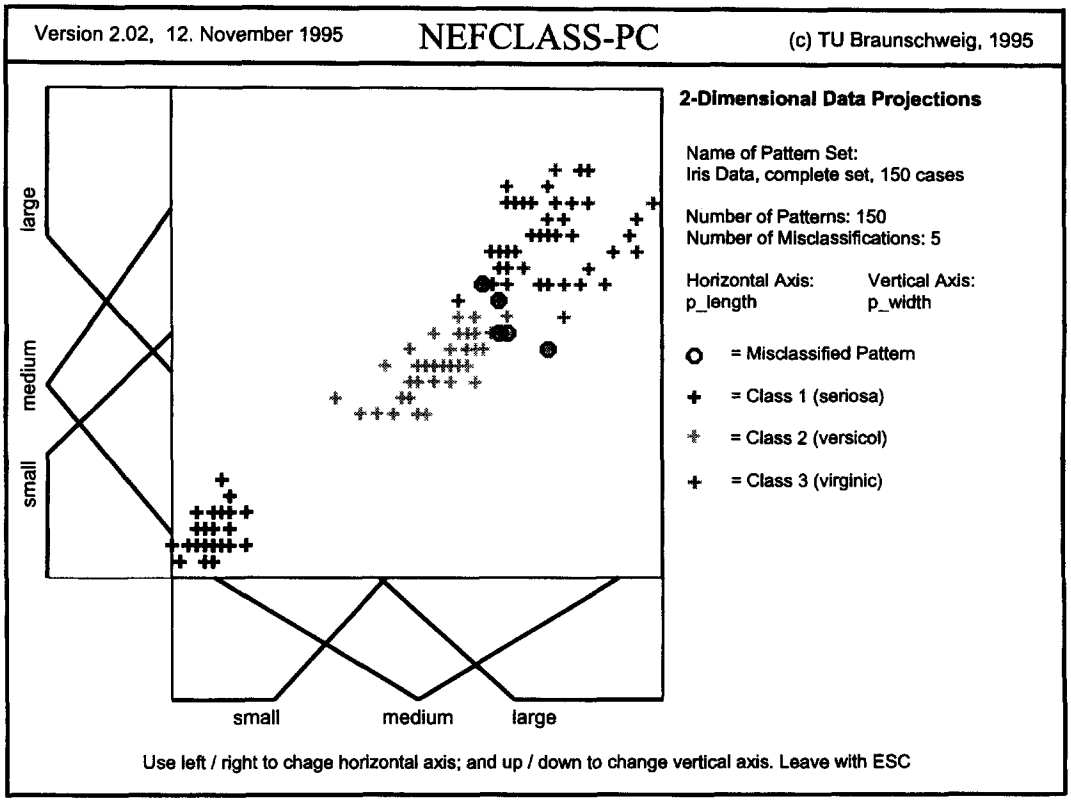

Fig. 4. A two-dimensional plot of the Iris data showing also the fuzzy sets learned by NEFCLASS. The system was initialized by three equally distributed fuzzy sets (shouldered or triangular) for each input feature. Only features x_3 and x_4 were used.

by Gustafson and Kessel [10] presented in [13]. This modification is called "GK parallel" and looks for axes parallel hyper-ellipsoids only, which results in a more efficient computation of clusters, and less loss of information, when fuzzy sets are constructed by projecting the clusters. We used the tool FCLUSTER to perform fuzzy clustering (Fig. 5).

FCLUSTER discovered three clusters that were interpreted as fuzzy rules by projecting the clusters to each dimension and finding trapezoidal membership functions that closely matched the projections. The membership functions were interpreted by *small*, *medium* and *large* resulting in three rules that caused 94 classification errors on the whole data set (see Table 3).

Feature x_9 has the same value in all rules, therefore we leave it out in the NEFCLASS system. When we initialize a NEFCLASS system with these three rules, we at first get 240 classification errors. However, after 80 epochs of training, we obtained a result of only 50 errors (92.7% correct).

Table 3
Fuzzy rules obtained by fuzzy clustering used to initialize a NEFCLASS system (s = small, m = medium, l = large)

R_1: **if** (s, s, s, s, s, s, s, s, s) **then** *benign*
R_2: **if** (m, m, m, m, m, l, m, m, s) **then** *malign*
R_3: **if** (m, l, l, l, m, l, m, l, s) **then** *malign*

Trying to use NEFCLASS without prior knowledge, and a maximum of three rules, we do not get a result by using "best" rule learning, because only one class is covered by the best three rules. Using four rules and "best per class" rule learning results in a NEFCLASS system performing badly with 135 errors (80.4% correct). So in this case using prior knowledge is a substantial advantage.

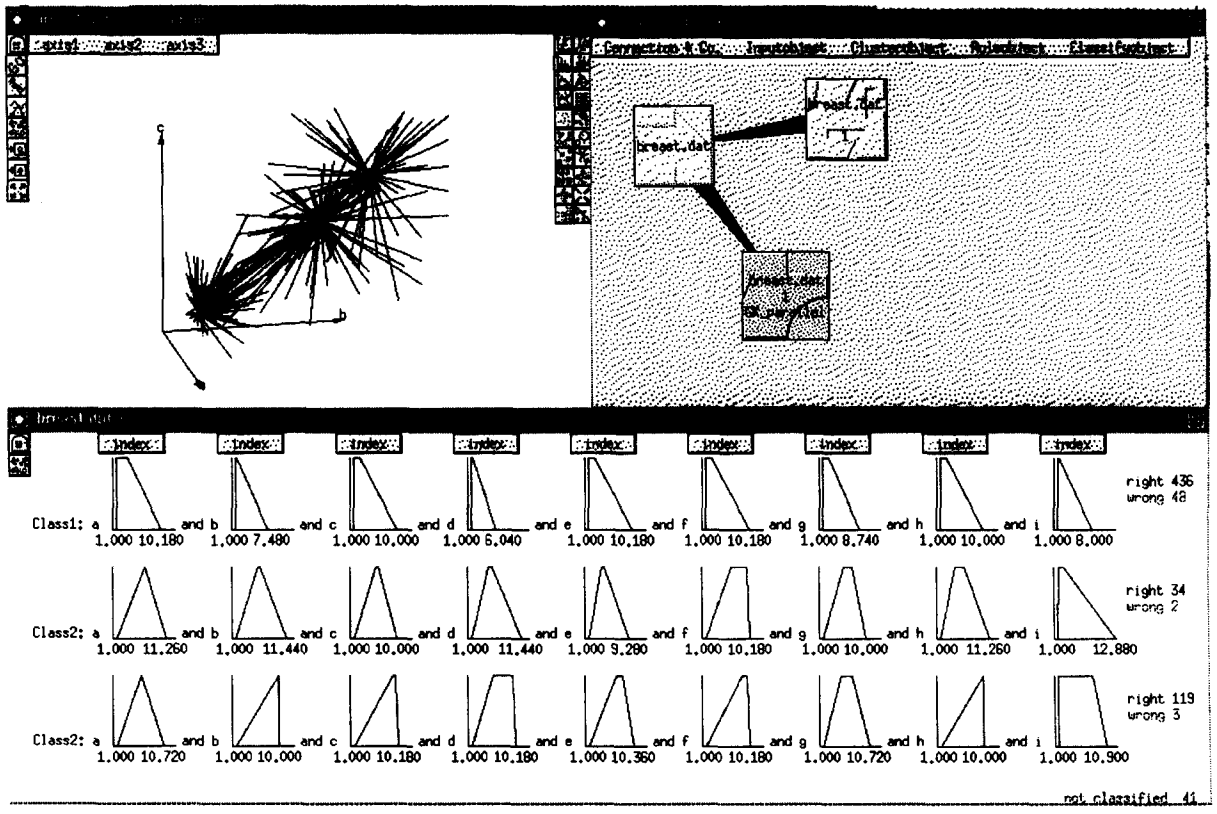

Fig. 5. Fuzzy rules for the "Wisconsin Breast Cancer Data" obtained by fuzzy clustering (GK parallel) using the toolFCLUSTER.

By analyzing the fuzzy sets obtained by training the three rules used as prior knowledge we found that the fuzzy set *medium* substantially overlapped with either the fuzzy set *small* or *large* for almost all variables. This can be seen as evidence that *medium* is superfluous, and therefore we again trained a NEFCLASS system with "best per class" rule learning, allowing it to create four rules. This time we only used two fuzzy sets to partition the domains of each variable. After 100 epochs of training NEFCLASS made only 24 errors (96.5% correct). It has found the rules shown in Table 4.

The examples with the Iris data and the Wisconsin Breast Cancer data show that NEFCLASS can be used as an interactive data analysis method. It is useful to provide prior knowledge when this is possible. A combination with fuzzy clustering can help here. The learning result of NEFCLASS should be analyzed, and the obtained information can be used for another run that yields an even better result.

Table 4
Fuzzy rules found by NEFCLASS, when only two fuzzy sets per input variable were used

R_1: **if** (s, s, s, s, s, s, s, s, s) **then** *benign*
R_2: **if** (l, s, s, s, s, s, s, s, s) **then** *benign*
R_3: **if** (l, l, l, l, l, l, l, l, s) **then** *malign*
R_4: **if** (l, l, l, l, s, l, l, l, s) **then** *malign*

5. Conclusions

We have discussed a neuro-fuzzy model for pattern classification that can learn fuzzy rules and membership functions from training data by a simple learning algorithm. The learning algorithm does not manipulate n-dimensional clusters in pattern space, but directly adapts membership functions over each single dimension. This method is computationally less expensive,

and can take constraints into account, which try to support the learning procedure to produce fuzzy rule bases that are more easy to interpret than rule bases induced by e.g. fuzzy clustering techniques.

Due to its orientation to semantics and interpretation of rules, the learning algorithm is less flexible in creating a rule base from training data alone than cluster algorithms. It is therefore useful to exploit the property of neuro-fuzzy models to integrate prior knowledge. If knowledge in form of fuzzy rules does not exist, it is possible to use fuzzy clustering to create an initial set of rules that is then completed and refined by the neuro-fuzzy learning algorithm.

The NEFCLASS software for MS-DOS PC that was used to obtain the results described in this paper can be freely obtained by anonymous ftp from fuzzy.cs.uni-magdeburg.de in /pub/local/nefclass, or from the World Wide Web (http: //fuzzy.cs.uni-magdeburg.de).

References

[1] S. Abe, M.-S. Lan, A method for fuzzy rules extraction directly from numerical data and its application to pattern classification, IEEE Trans. Fuzzy Systems 3 (1995) 18–28.

[2] S. Abe, M.-S. Lan, R. Thawonmas, Tuning of a fuzzy classifier derived from data, Int. J. Approximate Reasoning 14 (1996) 1–24.

[3] H.R. Berenji, P. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, IEEE Trans. Neural Networks 3 (1992) 724–740.

[4] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

[5] J.C. Bezdek, S.K. Pal (Eds.), Fuzzy Models for Pattern Recognition, IEEE Press, New York, 1992.

[6] J.C. Bezdek, E.C.-K. Tsao, N.R. Pal, Fuzzy Kohonen clustering networks, in: Proc. IEEE Int. Conf. on Fuzzy Systems, San Diego, 1992, pp. 1035–1043.

[7] J.J. Buckley, Y. Hayashi, Fuzzy neural networks: a survey, Fuzzy Sets and Systems 66 (1994) 1–13.

[8] J.J. Buckley, Y. Hayashi, Neural networks for fuzzy systems, Fuzzy Sets and Systems 71 (1995) 265–276.

[9] R. Fisher, The use of multiple measurements in taxonomic problems, Annual Eugenics 7, Part II (1936) 179–188.

[10] D. Gustafson, W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in: Proc. IEEE CDC, San Diego, 1979, pp. 761–766.

[11] S.K. Halgamuge, M. Glesner, Neural networks in designing fuzzy systems for real world applications, Fuzzy Sets and Systems 65 (1994) 1–12.

[12] J.M. Keller, H. Tahani, Backpropagation neural networks for fuzzy logic, Inform. Sci. 62 (1992) 205–221.

[13] F. Klawonn, R. Kruse, Constructing a fuzzy controller from data, Fuzzy Sets and Systems 85 (1997).

[14] F. Klawonn, D. Nauck, R. Kruse, Generating rules from data by fuzzy and neuro-fuzzy methods, in: Proc. Fuzzy-Neuro-Systeme '95, Darmstadt, 1995, pp. 223–230.

[15] R. Kruse, J. Gebhardt, F. Klawonn, Foundations of Fuzzy Systems, Wiley, Chichester, 1994.

[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, Building neural fuzzy controllers with NEFCON-I, in: R. Kruse, J. Gebhardt, R. Palm (Eds.), Fuzzy Systems in Computer Science, Vieweg, Braunschweig, 1994, pp. 141–151.

[18] D. Nauck, Fuzzy neuro systems: an overview, in: R. Kruse, J. Gebhardt, R. Palm (Eds.), Fuzzy Systems in Computer Science, Vieweg, Braunschweig, 1994, pp. 91–107.

[19] D. Nauck, F. Klawonn, R. Kruse, Foundations of Neuro-Fuzzy Systems, Wiley, Chichester, 1997, to be published.

[20] D. Nauck, R. Kruse, NEFCLASS – a neuro-fuzzy approach for the classification of data, in: K. George, J.H. Carrol, E. Deaton, D. Oppenheim, J. Hightower (Eds.), Applied Computing 1995, Proc. of the 1995 ACM Symp. on Applied Computing, ACM Press, New York, 1995, pp. 461–465.

[21] D. Nauck, R. Kruse, Designing neuro-fuzzy systems through backpropagation, in: W. Pedrycz (Ed.), Fuzzy Modelling: Paradigms and Practice, Kluwer, Boston, 1996, pp. 203–228.

[22] D. Nauck, R. Kruse, Neuro-fuzzy systems research and applications outside of Japan (in Japanese), in: M. Umano, I. Hayashi, T. Furuhashi (Eds.), Fuzzy-Neural Networks (in Japanese) Soft Computing Series. Asakura Publ., Tokyo, 1996, pp. 108–134.

[23] S. Pal, S. Mitra, Multi-layer perceptron, fuzzy sets and classification, IEEE Trans. Neural Networks 3 (1992) 683–697.

[24] N. Tschichold-Gürman, Generation and improvement of fuzzy classifiers with incremental learning using Fuzzy Rulenet, in: K. George, J.H. Carrol, E. Deaton, D. Oppenheim, J. Hightower (Eds.), Applied Computing 1995. Proc. of the 1995 ACM Symp. on Applied Computing, ACM Press, New York, February 1995, pp. 466–470.

[25] N. Tschichold-Gürman, RuleNet – a new knowledge-based artificial neural network model with application examples in robotics, Ph.D. Thesis, ETH Zürich, 1996.

[26] P. Vuorimaa, Fuzzy self-organizing map, Fuzzy Sets and Systems 66 (1994) 223–231.

[27] W. Wolberg, O. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, Proc. National Academy of Sciences, vol. 87, 1990, pp. 9193–9196.