Learning Methods for Fuzzy Systems

(Lernverfahren für Fuzzy–Systeme)

Rudolf Kruse and Detlef Nauck
Department of Computer Science
Technical University of Braunschweig
Bültenweg 74 / 75, D-38106 Braunschweig, Germany
Tel.: +49.531.391.3155, Fax: +49.531.391.5936, Email: nauck@ibr.cs.tu-bs.de

---

This paper is going to appear in the Proceedings of the 3rd German GI–Workshop "Neuro–Fuzzy-Systeme" to be held in Darmstadt, Germany, from November 15–17, 1995

Kurzfassung

In diesem Beitrag diskutieren wir einen Teilaspekt der Lernmethoden für Fuzzy–Systeme, die Semantik von Neuro–Fuzzy–Systemen. Ein Neuro–Fuzzy–System sollte stets so strukturiert sein, daß es ohne Schwierigkeiten zu interpretieren ist, und die verwendeten Lernverfahren sollten immer zu semantisch eindeutigen Fuzzy–Regeln führen. An einem Beispiel vergleichen wir die Ergebnisse eines Fuzzy–Clusteringverfahrens mit den Lernergebnissen eines Neuro–Fuzzy–Systems im Kontext der Musterklassifikation.

Abstract

In this paper we discuss on aspect on learning methods for fuzzy systems: the semantics of neuro–fuzzy systems. We show how a neuro–fuzzy system should be structured, so it can be easily interpreted, and how learning algorithms for these models can be constructed. Learning in neuro–fuzzy systems should always lead to interpretable fuzzy rules. As an example, we compare the fuzzy rules resulting from a fuzzy clustering procedure to the the learning results of a neuro–fuzzy system in the context of pattern classification.

Introduction

The importance and popularity of combinations between neural networks and fuzzy systems – so–called neuro–fuzzy systems – grew tremendously over the last few years (for an overview see [15, 17, 18]). Some of the early neuro–fuzzy systems are ad–hoc models with semantical flaws.

To provide formal foundations we presented a generic fuzzy perceptron [16, 21] that can be used to derive special neuro–fuzzy models that are suitable for neuro–fuzzy control [14, 19, 20], or neuro–fuzzy data analysis [22]. We argue, that a neuro–fuzzy system must always – before, during and after learning – be interpretable in terms of fuzzy if–then rules. This implies, that certain constraints must be applied to the structure of the neuro–fuzzy system, and to the learning algorithm that is used by the system.

There are several approaches to combine neural networks and fuzzy systems:

a) Concurrent Neural/Fuzzy Models: This is the weakest form of neuro–fuzzy combinations. A neural network and a fuzzy system are used concurrently on the same task. The neural network may be used as a pre– or post–processor to the fuzzy system, but the neural network does not change any parameters of the fuzzy system.

b) Cooperative Neuro–Fuzzy Models: In this case a neural network, or in most cases just a neural learning algorithm, is used to learn certain parameters (fuzzy sets, fuzzy rules, or rule weights) of the fuzzy system. After learning, the neural network is not present anymore, only the resulting fuzzy system is used.

c) Hybrid Neuro–Fuzzy Models: Here a new architecture is created that can be either interpreted as a neural network, or as a fuzzy system. Neural network and fuzzy system are not separable anymore. Modern neuro–fuzzy approaches are from this type.

In this paper we will only consider hybrid neuro–fuzzy architectures. In the following we give a short overview on different hybrid neuro–fuzzy models, and then we briefly present the fuzzy perceptron model. Using the models NEFCON and NEFCLASS, which can be derived from the fuzzy perceptron, as an example we then discuss some considerations on neuro–fuzzy learning algorithms. As an example on the fuzzy rules obtained from a neuro–fuzzy system, we compare the classification results of the NEFCLASS model, to a fuzzy clustering approach using the well known Iris data set [2].

Hybrid Neuro–Fuzzy Approaches

As discussed in the next section, we use a perceptron–like structure, that has fuzzy sets as weights, in contrast to other approaches were fuzzy and neural concepts are combined to perceptron–like architectures using "regular" weights, i.e. real values. In general the three following kinds of hybrid neuro–fuzzy models can be distinguished.

(i) *Neuro–fuzzy networks based on sampled fuzzy sets:* It is easily possible to create a feedforward neural network that receives fuzzy sets as inputs and produces fuzzy sets as outputs.

The fuzzy sets are usually sampled at a fixed number of values over their domains and represented by vectors of membership degrees. For each membership degree a single input or output neuron is used. The neural network is then trained or constructed in such a way that it represents a given set of fuzzy rules by mapping membership degrees entered in the neurons of the input layer to membership degrees in the neurons of the output layer. The network output is interpreted as a vector representation of an output fuzzy set. Two kinds of these approaches can be distinguished: One where the input/output behavior of fuzzy rules is approximated [8], [9] and another one, were a special network is constructed to implement fuzzy rules directly [10].

In the first approach a usual multilayer perceptron with the usual propagation and activation functions and backpropagation learning can be used. In this case only the input/output behavior of the network can be interpreted, but not its structure or weights. By just looking at the network, it is not possible to see which rules were used to train the system. The membership degrees are represented by the activations of the input and output units, and cannot be derived from the weights of the network. This means such kind of networks do not have clear semantics, like common neural networks they are just black boxes.

For the second approach special propagation and activation functions, and special interconnection schemes are used to implement fuzzy rules in a network structure by using vector representations of sampled fuzzy sets. To train this second kind of networks special learning algorithms must be constructed. Because the fuzzy rules are used to *construct* the network, it has clear semantics in the form of fuzzy if–then rules, and its structure is interpretable. The sampled membership degrees of the fuzzy sets are directly represented by the weights of the network. When a learning algorithm is applied, it must be carefully observed that the semantics of the network is not destroyed.

Approaches of both kind are not able to learn fuzzy rules (structure learning), at best they can learn the form of membership functions (second approach).

(ii) *Neuro–fuzzy systems using parameterized fuzzy sets stored in "neurons":* A lot of modern neuro–fuzzy approaches use parameterized membership functions that are stored within the "neurons" of a multilayered feedforward architecture, e.g. ANFIS [7], FUN [25], GARIC [1] and FuNe [6]. The membership functions are usually parameterized Gaussians, trapezoidal, or triangular functions, or they are constructed by superposing sigmoids (FuNe). The learning algorithms are usually backpropagation–like procedures. To be able to apply backpropagation all functions used within the system for the fuzzy inference process (membership functions, evaluation of rule antecedents, rule accumulation) have to be differentiable. This can be achieved by implementing a fuzzy system of the Sugeno type and using multiplication as a t–norm. Sometimes so–called "soft–min" and "soft–max" functions are used, to obtain differentiable functions, that are similar to the t–norm min and the t–conorm max. Approaches like ANFIS or GARIC cannot learn fuzzy rules, but only fuzzy sets. Approaches like Fun and FuNe are also capable of structure learning.

An approach that resembles radial basis function networks is Fuzzy RuleNet [26]. It does not directly store membership functions in its structure, but fuzzy rules and fuzzy sets can be extracted. Fuzzy RuleNet clusters data by overlapping hyperboxes, and can learn fuzzy sets and fuzzy rules.

(iii) *Neuro–fuzzy systems using fuzzy sets as weights:* Two modern neuro–fuzzy approaches that use fuzzy sets as weights are NEFCLASS [22] and NEFCON [14], [19], [20]. They are derived from the model of a generic fuzzy perceptron that defines the basic properties of this kind of neuro–fuzzy models. They are described in the following two sections.

The Fuzzy Perceptron

A generic fuzzy perceptron has the architecture of an usual multilayer perceptron, but the weights are modeled as fuzzy sets and the activation, output, and propagation functions are changed accordingly. The intention of this model is 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. The fuzzy perceptron was used to derive the NEFCON model [14, 19, 20] for neuro–fuzzy control applications, and the NEFCLASS model for neuro–fuzzy data analysis [22].

Definition 1 *A* **3–layer generic 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 $W : U \times U \rightarrow \mathcal{F}(\mathbb{R})$*, such that there are only connections* $W(u, v)$ *with* $u \in U_i,\ v \in U_{i+1} (i \in \{1, 2\})$ *(*$\mathcal{F}(\mathbb{R})$ *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 $u \in U_1 \cup U_2$*:*

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

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

$$\begin{aligned} A_u &: \mathcal{F}(\mathbb{R}) \rightarrow \mathcal{F}(\mathbb{R}), \\ a_u &= A_u(\mathrm{net}_u) = \mathrm{net}_u. \end{aligned}$$

(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 $u \in U_3$:

$$\begin{aligned} O_u &: \mathcal{F}(\mathbb{R}) \to \mathbb{R}, \\ o_u &= O_u(a_u) = \mathrm{DEFUZZ}_u(a_u), \end{aligned}$$

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 $u \in U_1$:

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

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

$$\begin{aligned} \mathrm{NET}_u &: (\mathbb{R} \times \mathcal{F}(\mathbb{R}))^{U_1} \to [0,1], \\ \mathrm{net}_u &= \mathop{\top}_{u' \in U_1} \{W(u',u)(o_{u'})\}, \end{aligned}$$

where $\top$ is a t–norm,

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

$$\begin{aligned} \mathrm{NET}_u &: ([0,1] \times \mathcal{F}(\mathbb{R}))^{U_2} \to \mathcal{F}(\mathbb{R}), \\ \mathrm{net}_u &: \mathbb{R} \to [0,1], \\ \mathrm{net}_u(x) &= \mathop{\perp}_{u' \in U_2} \{\top(o_{u'}, W(u',u)(x))\}, \end{aligned}$$

where $\perp$ 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

$$o_u = x, \ \text{with } net_u(x) = 1.$$

is used instead of (iv.b).

(vi) $\text{ex} : U_1 \to \mathbb{R}$, *defines for each input unit* $u \in U_1$ *its external input* $\text{ex}(u) = \text{ex}_u$. *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, and 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. Each hidden unit represents a fuzzy rule, where the weights coming from the input layer are the rule antecedents, and the weight(s) leading to the output layer are the rule conclusion(s).

Learning in Neuro–Fuzzy Systems

The generic fuzzy perceptron is usually not directly applicable. It could be applied from a computational point of view, but from a semantical point of view this would be not advisable. To be able to correctly interpret the structure of the fuzzy perceptron certain restrictions have to be used:

(i) The fuzzy weights must be implemented in such a way, that identical linguistic terms are represented by identical fuzzy sets at all times.

(ii) There must be no rule units with identical incoming fuzzy weights, but different outgoing weights, for this would constitute an inconsistent rule base.

(iii) In some cases it might be necessary, that each rule unit is connected to one and only one output unit (usually for classification tasks).

For different application areas different neuro–fuzzy models have to be derived from the generic fuzzy perceptron. To realize a neuro–fuzzy system for control purposes, where fuzzy sets are used to partition the input and output variables (e.g. like in a Mamdani controller), a NEFCON (NEuro–Fuzzy CONtrol) system [14, 18, 19, 20] can be derived:

Definition 2 *A NEFCON system is a 3–layer fuzzy perceptron with the following specifications:*

(i) The input units are denoted as $\xi_1, \ldots, \xi_n$, *the hidden rule units are denoted as* $R_1, \ldots, R_k$, *and the single output unit is denoted as* η.

(ii) Each connection between units ξ_i *and* R_r *is labeled with a linguistic term* $A^{(i)}_{j_r}$ $(j_r \in \{1, \ldots, p_i\})$.

(iii) Each connection between units R_r *and the output unit* η *is labeled with a linguistic term* B_{j_r} $(j_r \in \{1, \ldots, q\})$.

(iv) Connections coming from the same input unit ξ_i *and having identical labels, bear the same fuzzy weight at all times. These connections are called* **linked connections,** *and their weight is called* **shared weight**. *An analogous condition holds for the connections leading to the output unit* η.

(v) Let $L_{\xi,R}$ *denote the label of the connection between the input unit* ξ *and the rule unit* R. *For all rule units* R, R', $(\forall \xi \; L_{\xi,R} = L_{\xi,R'}) \Longrightarrow R = R'$ *holds.*

To realize a neuro–fuzzy system for classification, where each pattern belongs to exactly one class, a NEFCLASS (NEuro–Fuzzy CLASSification) system [22] can be derived from the generic fuzzy perceptron:

Definition 3 *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* $A^{(i)}_{j_r}$ $(j_r \in \{1, \ldots, q_i\})$.

(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:*

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

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

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

(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:*

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

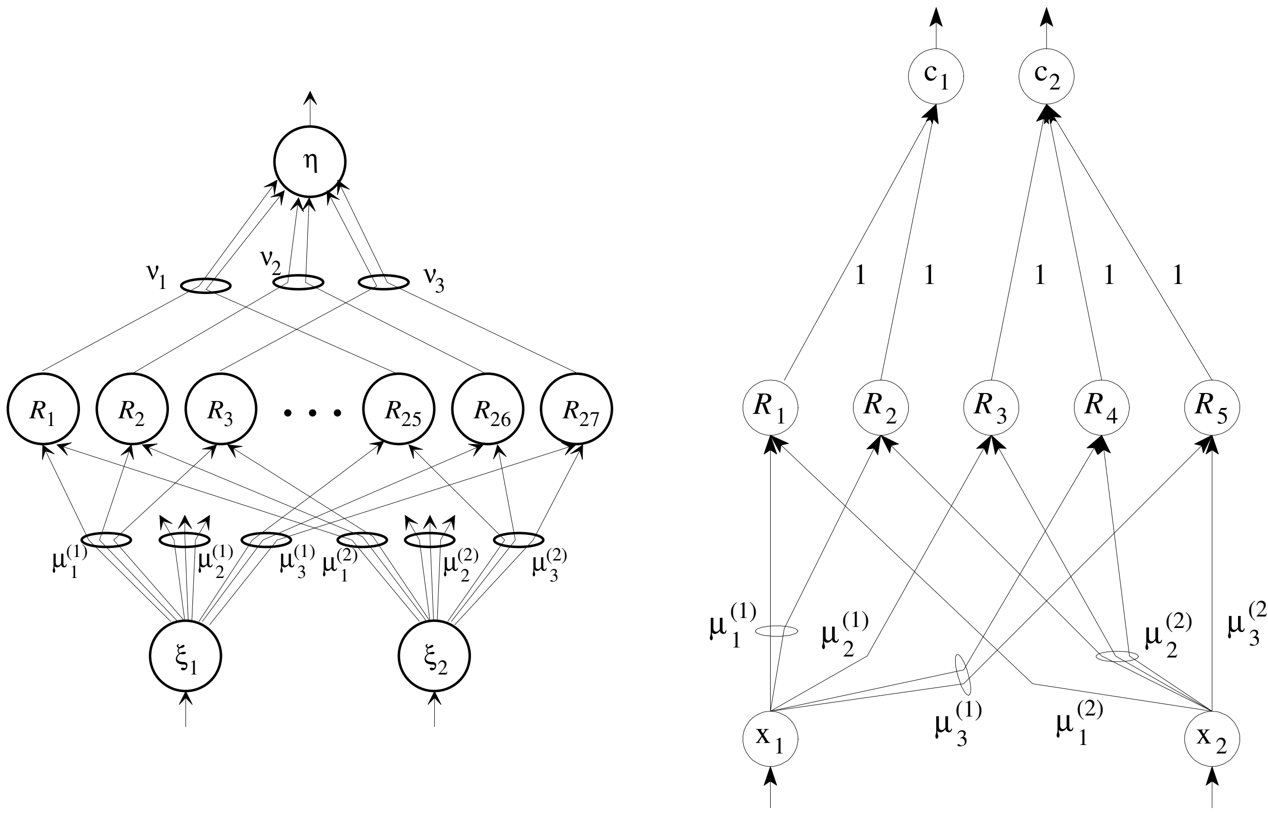

Figure 1: Two neuro–fuzzy models derived from the generic fuzzy perceptron: NEFCON (left) for control purposes learning by reinforcement, and NEFCLASS (right) for data analysis using a supervised learning algorithm (the ellipses represent shared weights)

The two models defined above are shown in Fig. 1. Both definitions make it possible to interpret NEFCON and NEFCLASS systems in terms of a fuzzy system; each hidden unit represents a fuzzy if-then rule. Condition (iv) specifies in both definitions that there have to be *shared weights*. If this feature is missing, it would be possible for fuzzy weights representing identical linguistic terms to evolve differently during the learning process. If this is allowed to happen, the architecture of the neuro–fuzzy systems cannot be understood as fuzzy rule bases. Condition (v) determines in both definitions that there are no rules with identical antecedents.

To implement a learning algorithm for a neuro–fuzzy system typically derivations of the backpropagation learning rule are used. As already mentioned, a direct application of backpropagation would imply that all functions used within the neuro–fuzzy model are differentiable. For approaches derived from the generic fuzzy perceptron this is usually not the case, because the t–norm min is used to evaluate the rule antecedents, and the function used to accumulate all rule outputs can also be non–differentiable. In addition to this triangular membership functions are used which have three non–differentiable points.

However it is possible to realize a simple heuristic, that uses the main idea of backpropagation: Determine an error value, propagating it back, spreading it through the architecture, and using it to locally cause adaptions in the system parameters. If it is possible to calculate an error value due to the performance of the neuro–fuzzy system, and the individual error of each rule encoded within the system, then it is also possible to use this information to change the fuzzy sets (or fuzzy rules) of the system.

The changes in the parameters are realized by a rewarding/punishing scheme. If a rule behaves erroneous, it has to be punished, i.e. its influence has to be reduced, and if it behaves properly, its influence can be enhanced. This change in influence can be reached by shifting the fuzzy sets or by enlarging/reducing their support.

By this learning technique the semantics of the neuro–fuzzy system can be easily conserved. Each membership function will only be changed, if a set of given constraints is not violated by the respective change. This set of constraints can be defined in such a way, that the learning process does always result in an interpretable fuzzy rule base.

Rule learning can be done in two ways: incrementally and decrementally. Incrementally learning fuzzy rules means to start with an empty rule base (no hidden units) and add rules whenever the system cannot cope with the input data. This can usually only be done if there is proper information about the desired output data (i.e. supervised learning). Decrementally learning rules means to start with an overconstrained and therefore inconsistent system, and making it consistent by deleting superfluous rules. This is computationally expensive, but it can be done if there is no information on the desired output, i.e. when reinforcement learning is used.

For NEFCLASS a supervised learning procedure with incremental rule learning is used [22], and for NEFCON a reinforcement learning algorithm with decremental rule learning can is applied [14, 18, 19, 20].

A neuro–fuzzy system is supposed to provide an approximate solution. The error measure of the learning algorithm should take this into account. For the neuro–fuzzy models derived from the generic fuzzy perceptron a fuzzy error measure was defined [19]. For reinforcement learning situations, where the desired output value is not known, the performance of the system can be described by a fuzzy rule base that determines a fuzzy error value for the current state. Compared to the usual binary reinforcement signal (failed, not failed) used for neural networks this provides more information to the learning algorithm, resulting in faster adaption. This way a fuzzy error does not only tolerate near optimal solutions (i.e. solutions with a “small error”), but also exploits additional knowledge about the task to be solved. The way the fuzzy error rule base is defined also allows incremental rule learning for an enhanced NEFCON learning procedure [23].

For supervised learning processes the output value for a given input is known, and an exact error value can be obtained. For neuro–fuzzy systems it is useful to define one or more fuzzy sets describing the possible error values, so the learning algorithm can be more tolerant about small deviations from the optimal solution [16, 22]. This can be especially beneficial when the training data is noisy, and overfitting must be avoided.

It must be clear that the above mentioned neuro–fuzzy models (and in fact almost all neuro–fuzzy models described in literature) have nothing to do with fuzzy logic in the narrow sense. A neuro–fuzzy system is used to approximate a unknown function, that is described by a number of vague points represented by fuzzy if–then rules. So if – like in the models derived from the generic fuzzy perceptron – proper precautions are taken in constructing the neuro–fuzzy system, and in training it, then its semantics is identical to the semantics of a fuzzy controller.

When the domain requires approximate reasoning methods, and simple function approximation is not sufficient, then graphical models based on possibilistic logic can be used. In this area there also exist learning algorithms to extract possibilistic dependency networks from data [4, 5].

Deriving Fuzzy Rules From Data

In this section we will compare the results of a NEFCLASS system to the results of a fuzzy clustering procedure. The task is to obtain classification rules to correctly classify the patterns of the well known Iris data set [2]. 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 domain, and it is often used as a benchmark.

The patterns are vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and a class C is a (crisp) subset of $\mathbb{R}^n$. The intersection of two different classes is 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 fuzzy sets, and the rule base contains k fuzzy rules $R_1, \ldots, R_k$.

The task is to find fuzzy rules of the form:

$$\begin{array}{ll} R\text{:} & \textbf{if } x_1 \text{ is } \mu_1 \textbf{ and } x_2 \text{ is } \mu_2 \textbf{ and } \ldots \textbf{ and } x_n \text{ is } \mu_n \\ & \textbf{then} \text{ the pattern } (x_1, x_2, \ldots, x_n) \text{ belongs to class } i, \end{array}$$

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 rule base of NEFCLASS is an approximation of an (unknown) function $\varphi : \mathbb{R}^n \to \{0,1\}^m$ that represents the classification task where $\varphi(\mathbf{x}) = (c_1, \ldots, c_m)$

such that $c_i = 1$ and $c_j = 0$ ($j \in \{1, \ldots, m\}, j \neq i$), i.e. $\mathbf{x}$ 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 $\varphi(\mathbf{x})$ by $\varphi(\mathbf{x}) = \psi(\varphi'(\mathbf{x}))$, 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 $\mathbf{c}$ to 1 and its other components to 0, respectively (winner–take–all principle).

For the fuzzy clustering procedure a variation of an algorithm designed by Gath and Geva [3] was used. This variation provides much better results than standard FCM clustering, and it is optimized for fuzzy rule generation [11]. The rules derived from the clustering result are also evaluated using the winner–take–all principle.

IF V_1 is small	and V_2 is medium	and V_3 is small	and V_4 is small	THEN Cl. 1
IF V_1 is large	and V_2 is medium	and V_3 is medium	and V_4 is medium	THEN Cl. 2
IF V_1 is small	and V_2 is small	and V_3 is medium	and V_4 is medium	THEN Cl. 2
IF V_1 is medium	and V_2 is medium	and V_3 is large	and V_4 is large	THEN Cl. 3
IF V_1 is medium	and V_2 is small	and V_3 is large	and V_4 is large	THEN Cl. 3
IF V_1 is large	and V_2 is small	and V_3 is large	and V_4 is large	THEN Cl. 3
IF V_1 is large	and V_2 is medium	and V_3 is large	and V_4 is large	THEN Cl. 3

Table 1: The rules to classify the Iris data set learned by NEFCLASS

For the NEFCON learning process the data set was split in half, and the patterns were ordered alternately within the training and test data sets. We let the system create a maximum of seven rules. 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 75 patterns incorrectly (i.e. 97.3% correct). Considering all 150 patterns the system performed well with 96.67% correct classifications.

In Table 1 the fuzzy rules learned by NEFCLASS are shown. NEFCLASS started with no rule in the beginning, and created one rule to manage the first and easiest class. Two classify the patterns of the second and third class, two and four rules were created. After rule creation the initial membership functions were optimized to reduce the number of classification errors.

In Fig. 2 the fuzzy sets learned by NEFCLASS are shown. The system is initialized by three equally distributed fuzzy sets (shouldered or triangular) for each input feature. The learning result shows, that the fuzzy sets labeled with *medium* are obviously superfluous for the first two variables, and a partitioning of the domains into *small* and *large* would be sufficient. The membership function of the third and fourth variable have been corrected only a little bit.

Gath and Geva [3] designed fuzzy clustering methods that are looking for arbitrary hyper–

Figure 2: The fuzzy sets for the Iris data set learned by NEFCLASS

ellipsoidal clusters of varying size, whereas for rule induction it would be more advantageous to have axes–parallel hyper–ellipsoids. Therefore for clustering the Iris data set a variation of the Gath and Geva algorithm was used that was developed in [11].

The membership functions would be correctly determined by projecting the clusters. But this method is computational inefficient, and therefore, we use an approximation of this fuzzy set by projecting only the data set, and approximating this (discrete) projected fuzzy set by a trapezoidal or triangular membership function as for instance proposed in [24]. For the clustering process all 150 patterns were used together.

The results obtained by the fuzzy clustering procedure are shown in Fig. 3. This rule base produces seven errors (i.e. 95.3% correct) on the whole data set: three misclassifications, and four patterns that are not classified at all due to loss of information caused by the approximation of the membership functions. Compared to the rule base of NEFCLASS it is clear, that the rule base obtained by the fuzzy clustering procedure is less satisfying. The fuzzy sets cannot be as easily labeled as it is the case for the NEFCLASS solution, and even though there is one more rule, there are two more classification errors.

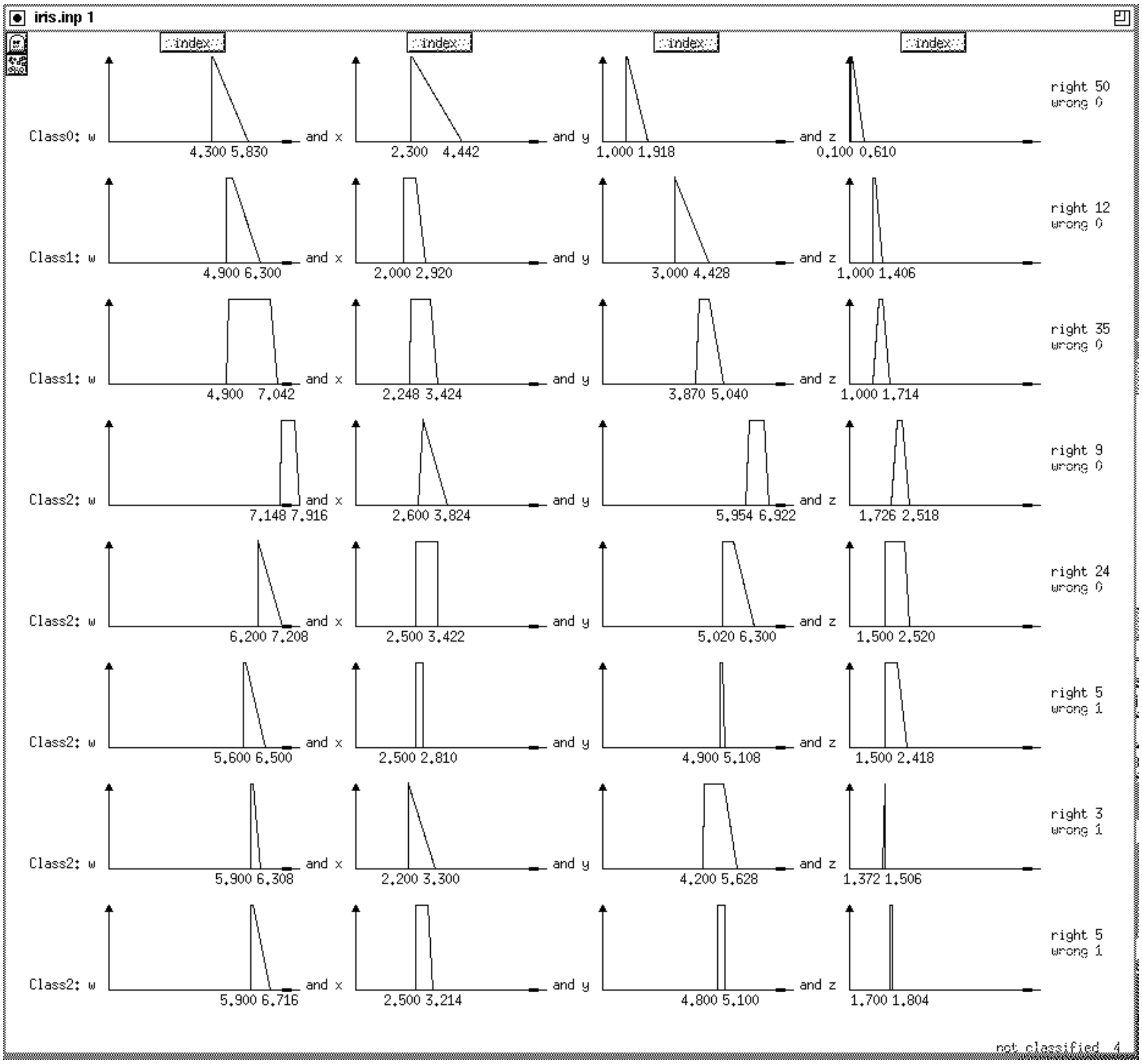

Figure 3: Classification rules for the Iris data set determined by fuzzy clustering

Even though a sophisticated fuzzy clustering procedure usually produces better classification results as a NEFCLASS system, this advantage is reduced, once the derivation of interpretable fuzzy rules is of interest. By approximating fuzzy sets a loss of information may occur, resulting in patterns that are not classified by any of the rules, and the classification performance becomes worse. The advantage of a NEFCLASS system is that it enhances its performance by directly manipulating given initial fuzzy partitionings. It is easily possible to define constraints for the learning algorithm like “fuzzy sets may not pass each other”, “the support of a fuzzy set may not collapse to a singleton”, etc. This way the learning result is more likely to be interpretable, and the semantics of the linguistic terms assigned to the fuzzy sets is conserved. There is no need to guess after learning which fuzzy set should be labeled by which linguistic term, and the problem that there are similar but different fuzzy sets in fact describing the same linguistic term does not exist.

As a result we can say, that approaches like NEFCLASS are preferable from a semantical point of view, because the resulting fuzzy rules are more likely to be interpretable. Fuzzy clustering usually provides better classifications, when only the cluster information is needed, and it is therefore preferable for its performance. But once fuzzy rules have to be derived the performances usually gets worse, and the resulting rules are not satisfactory. The question how a fuzzy clustering procedure could be used to initialize a NEFCLASS system is addressed in [12].

Conclusions

We have shown how neuro–fuzzy models can be derived from a generic fuzzy perceptron such that they display clear semantics. Properly constructed neuro–fuzzy systems perform function approximation by using interpretable fuzzy if–then rules. We have tested the performance of the NEFCLASS model against a fuzzy clustering procedure on the IRIS data set, and found that the fuzzy rules produced by the neuro–fuzzy approach are easier to interpret due to a constraint learning algorithm.

Further research will be conducted on neuro–fuzzy classification initialized by fuzzy clustering, and on the integration of boolean input variables, and the handling of missing values, which is important for the processing of e.g. medical data.

For the reader who is interested in testing the NEFCON and NEFCLASS models, we offer NEFCON–I for Unix systems, and NEFCLASS–PC for MS-DOS personal computers. The programs can be obtained from ftp.ibr.cs.tu-bs.de in the directory /pub/local/nefcon by anonymous ftp, or from the World Wide Web (http://www.cs.tu-bs.de/projects/nefcon/).

References

[1] Hamid R. Berenji and Pratap Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Trans. Neural Networks*, 3:724–740, September 1992.

[2] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(Part II):179–188, 1936. (also in: Contributions to Mathematical Statistics, Wiley, New York, 1950).

[3] I. Gath and A.B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11:773–781, 1989.

[4] Jörg Gebhardt and Rudolf Kruse. Focusing and learning in possibilistic dependencyy networks. In *Postproceedings of 2nd Gauss Symposium (Conference B: Statistical Sciences)*, Berlin, 1994. De Gruyter.

[5] Jörg Gebhardt and Rudolf Kruse. Learning possibilistic networks from data. In *Proc. of Fifth Int. Workshop on Artificial Intelligence and Statistics*, pages 233–244, Fort Lauderdale, Florida, 1994.

[6] Saman K. Halgamuge and Manfred Glesner. Neural networks in designing fuzzy systems for real world applications. *Fuzzy Sets and Systems*, 65:1–12, 1994.

[7] J.-S. Roger Jang. ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Trans. Systems, Man & Cybernetics*, 23:665–685, 1993.

[8] James M. Keller. Experiments on neural network architectures for fuzzy logic. In [13], pages 201–216, 1991.

[9] James M. Keller and Hossein Tahani. Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks. *Int. J. Approximate Reasoning*, 6:221–240, February 1992.

[10] James M. Keller, Ronald R. Yager, and Hossein Tahani. Neural network implementation of fuzzy logic. *Fuzzy Sets and Systems*, 45:1–12, 1992.

[11] F. Klawonn and R. Kruse. Constructing a fuzzy controller from data. *Fuzzy Sets and Systems*, 1995. To appear.

[12] Frank Klawonn, Detlef Nauck, and Rudolf Kruse. Generating rules from data by fuzzy and neuro–fuzzy methods. In *Proc. Neuro–Fuzzy–Systeme'95*, Darmstadt, November 1995.

[13] Robert N. Lea and James Villareal, editors. *Proc. of the Second Joint Technology Workshop on Neural Networks and Fuzzy Logic*, Lyndon B. Johnson Space Center, Houston, Texas, February 1991. NASA.

[14] Detlef Nauck. Building neural fuzzy controllers with NEFCON–I. In Rudolf Kruse, Jörg Gebhardt, and Rainer Palm, editors, *Fuzzy Systems in Computer Science*, Artificial Intelligence, pages 141–151. Vieweg, Wiesbaden, 1994.

[15] Detlef Nauck. Fuzzy neuro systems: An overview. In Rudolf Kruse, Jörg Gebhardt, and Rainer Palm, editors, *Fuzzy Systems in Computer Science*, Artificial Intelligence, pages 91–107. Vieweg, Wiesbaden, 1994.

[16] Detlef Nauck. A fuzzy perceptron as a generic model for neuro–fuzzy approaches. In *Proc. Fuzzy–Systeme'94*, Munich, October 1994.

[17] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. Combining neural networks and fuzzy controllers. In Erich Peter Klement and Wolfgang Slany, editors, *Fuzzy Logic in Artificial Intelligence (FLAI93)*, pages 35–46, Berlin, 1993. Springer–Verlag.

[18] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Neuronale Netze und Fuzzy–Systeme*. Vieweg, Wiesbaden, 1994.

[19] Detlef Nauck and Rudolf Kruse. A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. In *Proc. IEEE Int. Conf. on Neural Networks 1993*, pages 1022–1027, San Francisco, March 1993.

[20] Detlef Nauck and Rudolf Kruse. NEFCON–I: An X–Window based simulator for neural fuzzy controllers. In *Proc. IEEE Int. Conf. Neural Networks 1994 at IEEE WCCI'94*, pages 1638–1643, Orlando, June 1994.

[21] Detlef Nauck and Rudolf Kruse. Designing neuro–fuzzy systems through backpropagation. In Witold Pedrycz, editor, *Fuzzy Modelling: Paradigms and Practice*. Kluwer, 1995. To be published.

[22] Detlef Nauck and Rudolf Kruse. NEFCLASS – a neuro–fuzzy approach for the classification of data. In K.M. George, Janice H. Carrol, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Applied Computing 1995. Proc. of the 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 461–465. ACM Press, New York, February 1995.

[23] Detlef Nauck, Rudolf Kruse, and Roland Stellmach. New learning algorithms for the neuro–fuzzy environment NEFCON–I. In *Proc. Neuro–Fuzzy–Systeme'95*, Darmstadt, November 1995.

[24] M. Sugeno and T. Yasukawa. A fuzzy–logic–based approach to qualitative modeling. *IEEE Trans. Fuzzy Systems*, 1:7–31, 1993.

[25] Sandra M. Sulzberger, Nadine N. Tschichold-Gürman, and Sjur J. Vestli. Fun: Optimization of fuzzy rule based systems using neural networks. In *Proc. IEEE Int. Conf. on Neural Networks 1993*, pages 312–316, San Francisco, March 1993.

[26] Nadine Tschichold-Gürman. Generation and improvement of fuzzy classifiers with incremental learning using fuzzy rulenet. In K.M. George, Janice H. Carrol, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Applied Computing 1995. Proc. of the 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 466–470. ACM Press, New York, February 1995.