NEFCLASS — A NEURO–FUZZY APPROACH FOR THE CLASSIFICATION OF DATA

Detlef Nauck and Rudolf Kruse
Technical University of Braunschweig, Dept. of Computer Science

Keywords

Data analysis, learning, fuzzy system, neuro–fuzzy system, pattern recognition

Abstract

In this paper we present NEFCLASS, a neuro–fuzzy system for the classification of data. This approach is based on our generic model of a fuzzy perceptron which can be used to derive fuzzy neural networks or neural fuzzy systems for specific domains. The presented model derives fuzzy rules from data to classify patterns into a number of (crisp) classes. NEFCLASS uses a supervised learning algorithm based on fuzzy error backpropagation that is used in other derivations of the fuzzy perceptron.

Introduction

Combinations of neural networks and fuzzy systems are very popular (for an overview see [4, 6]), but most of the approaches are not easy to compare because they use very different architectures, activation functions, propagation and learning algorithms, etc. In [5] we presented a fuzzy perceptron as a generic model of multilayer fuzzy neural networks. It can be used as a common base for neuro–fuzzy architectures in order to ease the comparision of different approaches. By applying additional constraints to the definition of the fuzzy perceptron one can e.g. obtain a structure that can be interpreted as a usual fuzzy controller, and easily create a neuro–fuzzy controller this way [3, 8, 9].

In this paper we present an approach to neuro–fuzzy data analysis. The goal is to derive fuzzy rules from a set of data that can be separated in different crisp classes, i.e. at this moment we do not consider data where the patterns belong to overlapping or fuzzy categories. The fuzziness involved is due to an imperfect or incomplete measurement of features thus rendering it difficult to assign a pattern to the correct category.

This paper appears in: George, K.M., Carrol, Janice H., Deaton, Ed, Oppenheim, Dave, and Hightower, Jim (eds.): Applied Computing 1995. Proc. of the 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28. ACM Press, 1995.
The authors can be contacted at:
Technical University of Braunschweig, Department of Computer Science, Bueltenweg 74 – 75, D–38106 Braunschweig, Germany
Tel: +49.531.391.3155, Fax: +49.531.391.5936
email: nauck@ibr.cs.tu-bs.de
WWW: http://www.cs.tu-bs.de/~nauck

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 the pattern $(x_1, x_2, \ldots, x_n)$ belongs to class i,
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.

We will first briefly present the fuzzy perceptron model in section II, and in section III we show how the NEFCLASS model is derived from it. We also present the supervised learning algorithm. In the fourth section we discuss the learning results we got by applying NEFCLASS to the IRIS data set, and we compare the results to other approaches.

The Fuzzy Perceptron

A fuzzy perceptron has the architecture of an usual multilayer perceptron, but the weights are modelled 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. In [5] we suggested a generic model for fuzzy neural networks based on a 3–layer fuzzy perceptron. By using it to derive neural 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 [3, 8, 9] for neuro–fuzzy controll applications, and it is now used to define the NEFCLASS model discussed in this paper. We will therefore shortly present the definition of the generic fuzzy perceptron.

Definition 1 *A* **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 $W : U \times U \to \mathcal{F}(\mathbb{R})$, *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* $u \in U_1 \cup U_2$*:*

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

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

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

(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* $u \in U_3$*:*

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

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* $u \in U_1$*:*

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

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

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

where $\top$ *is a t–norm,*

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

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

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*

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

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, the net inputs, and the activations of the output units are modelled 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.

The NEFCLASS model

NEFCLASS means **NE**uro **F**uzzy **CLASS**ification, and is used to determine the correct class or category of a given input pattern. The patterns are vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and a class C is a (crisp) subset of $\mathbb{R}^n$. 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 $R_1, \ldots, R_k$.

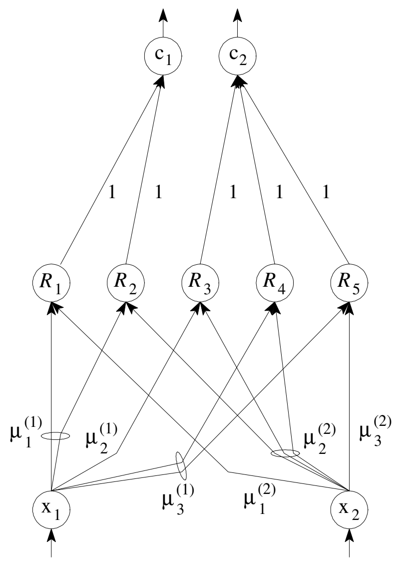

Figure 1: A NEFCLASS system with two inputs, five rules and two output classes

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

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. The 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 [7]. Connections that share a weight always come from the same input unit.

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 labelled 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 comming 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:*

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

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 the largest number of rule nodes that may be created in the hidden layer.

Each fuzzy set $\mu_j^{(i)}$ is labelled with a linguistic term $A_j^{(i)}$. This may be terms like *small, medium, large* etc. The fuzzy sets of those connections leading to the same rule unit R are also called the *antecedents* of R.

A NEFCLASS system that is created from scratch starts with no hidden units at all. They are created during a first run through the learning task $\tilde{\mathcal{L}}$, i.e. the set of examples. A rule is created by finding for a given input pattern $\mathbf{p}$ the combination of fuzzy sets, where each yields the highest degree of membership for the respective input feature. If this combination is not identical to the antecedents of an already existing rule, and if the permitted number of rule units is not yet reached, a new rule node is created. This is a very simple way of finding fuzzy rules, and it can only be successful, if the patterns are selected randomly from the learning task, and if the cardinalities of the classes are approximately equal. Another way would be, to use a scoring system, do not restrict the number of rules at first, and then later select those rules that have the highest scores. But from an implementational point of view, this would be more demanding, especially when problems with a large number of input features are considered.

When the rule base is created, the learning algorithm will adapt the membership functions of the antecedents. In this paper we consider triangular membership functions with three parameters:

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

We also use min as the t–norm to determine the degree of fulfillment of a rule, i.e. the activation of a rule node.

Definition 3 (NEFCLASS learning algorithm) *Consider a NEFCLASS system with* n *input units* $x_1, \ldots, x_n$, $k \leq k_{\max}$ *rule units* $R_1, \ldots, R_k$, *and* m *output units* $c_1, \ldots, c_m$. *Also given is a learning task* $\tilde{\mathcal{L}} = \{(\mathbf{p}_1, \mathbf{t}_1), \ldots, (\mathbf{p}_s, \mathbf{t}_s)\}$ *of* s *patterns, each consisting of an input pattern* $\mathbf{p} \in \mathbb{R}^n$, *and a target pattern* $\mathbf{t} \in \{0,1\}^m$. *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 $(\mathbf{p}, \mathbf{t})$ *from* $\tilde{\mathcal{L}}$

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

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

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

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

than create such a node, and connect it to the ouput 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.*

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

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

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

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

(a) *Determine the delta value*

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

(b) *Find x' such that*

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

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

$$\begin{aligned} \delta_b &= \sigma \cdot \delta_R \cdot (c - a) \cdot \operatorname{sgn}(o_{x'} - b), \\ \delta_a &= -\sigma \cdot \delta_R \cdot (c - a) + \delta_b, \\ \delta_c &= \sigma \cdot \delta_R \cdot (c - a) + \delta_b, \end{aligned}$$

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).*

Remarks

- The factor $o_R(1 - o_R)$ in step (iii.a) of the fuzzy set learning algorithm makes sure that the changes in the fuzzy weights are bigger, if a rule node has an activation of approximately 0.5, and that they are smaller if the activation approaches 0 or 1. By this a rule is "forced to decide", whether it wants to support a pattern or not.

- The sum in step (iii.a) is not really necessary, but allows to implement learning weights between rule and output units.

- The set of constraints Φ mentioned in step (iii.c) usually makes sure that the fuzzy sets keep their triangular form, and do not leave the domain of the respective input variable x.

- The end criterion that terminates the learning process is not easy to formulate, because the error can ususally not assume 0 due to the definition of net_c. A solution would be to define a maximum number of admissable misclassifications. ◇

Results and Semantical Aspects

In this section we present classification results obtained from NEFCLASS. We will also compare our findings with the performance of a common multilayer perceptron on the same data set, and with another neuro–fuzzy approach called FuNe I [2].

As an example we applied NEFCLASS to the IRIS data set [1]. 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 10 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.

We compared the learning results of the NEFCLASS system to an usual 3–layer perceptron trained by standard backpropagation. We used a network of 4 input, 3 hidden and 3 output units. The network was able to learn the training set completely, i.e. there were no classification errors on the 75 training patterns. But the network classified 5 of the 75 test patterns incorrectly, using the same criterion for classification as we did for the NEFCLASS system. So considering all 150 patterns both models performed equally well.

Incrementing the number of hidden units within the neural network caused more training cycles, and ususally did not produce a better result. We found one network with 10 hidden units that classified only 3 patterns of the test set incorrectly, but the 10 hidden units are not necessary to solve the problem, as can be seen in the before mentioned network with only 3 hidden units. We did not further try to obtain a better learning result with this network, e.g. by changing the learning parameters.

To obtain exact ouput values of 0 or 1, it would be necessary to learn the weights of the connections from the rule to output units, too. This would result in weighted rules, which give rise to some semantical problems, and we have addressed this question in several of our papers already [6, 7, 8, 9]. Allowing the weights to be selected from $[0, 1]$ could be interpreted as something like a degree of support for a rule. A value less than 1 would denote something like an ill–defined rule, that supports a class only to some extent. But one would still not receive output values near 1, they would be even smaller than with the weights fixed at 1. We tried to learn the weights within $[0, 1]$ and found that they usually have values greater than 0.95, and that the learning results did not benefit from this. Another approach would be to allow the weights to assume any value in $\mathbb{R}$, but one would leave the semantics of fuzzy rules behind, because it is not clear how rules weighted by absolute values greater than 1 or by negative values should be interpreted (for rules with negative weights sometimes an interpretation as *if not* rules is suggested [2]). Therefore we completely refrained from learning the weights in such a way. But if an exact output value is needed, and the interpretation of the learning result is not important, this approach can make sense, anyway.

Another neuro–fuzzy approach to pattern classification is presented in [2]. The authors use a system called "FuNe I" to obtain rules and fuzzy sets from data samples. The authors of [2] have applied their FuNe I system to several real world examples, and also used the IRIS data set split in a training and a test set. They have reached a classification rate of 99% on the test set 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 [2], 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 if FuNe I has a much more complex structure, and a more elaborated training procedure.

We think that simple models like NEFCLASS that are easy to interpret should be tried first before turning over to more complex approaches like FuNe I. They are usually harder to interpret, and need more expensive training algorithms. In addition there are sometimes semantical problems. FuNe I, for example, can create non–normal fuzzy sets, and rule weights greater than 1 (which is necessary to obtain exact classifications, i.e. nearly exact output values). This makes the interpretation of the model in form of fuzzy rules more difficult. But approaches like FuNe I can be useful, when models like NEFCLASS fail.

Conclusions

In this paper we have presented the NEFCLASS model, a neuro–fuzzy classification system derived from the generic model of a 3–layer fuzzy perceptron. NEFCLASS can be initialized by prior knowledge using fuzzy if–then rules and it can also be interpreted this way after the learning process, i.e. it is not a black box like usual neural networks. The model is able to create fuzzy rules from data examples by a single run through the data set. After creating a rule base NEFCLASS learns its fuzzy sets by adapting parameters of the membership functions in a supervised learning algorithm. The rules are not weighted like in other neuro–fuzzy approaches, thus avoiding semantical problems, and simplifying the interpretation of the learning result.

NEFCLASS was tested on the IRIS data set and the performance was satisfactory compared to usual neural networks and another neuro–fuzzy approach FuNe I. Further research will be conducted on applying NEFCLASS to real world classification problems, 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. An implementation of a more elaborated rule learning algorithm will also be considered.

For the reader who is interested in testing the NEFCLASS model or in repeating our learning results we offer NEFCLASS–PC for MS-DOS personal computers. The programm and the necessary data files can be obtained (from January 1995 on) by anonymous ftp from ftp.ibr.cs.tu-bs.de in the directory /pub/local/nefclass, or from the World Wide Web (http://www.cs.tu-bs.de/~nauck).

References

[1] 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).

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

[3] 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.

[4] 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.

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

[6] 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.

[7] Detlef Nauck and Rudolf Kruse. Interpreting changes in the fuzzy sets of a self-adaptive neural fuzzy controller. In *Proc. Second Int. Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems (IFIS'92)*, pages 146–152, College Station, Texas, December 1992.

[8] 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.

[9] 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.