New Learning Strategies for NEFCLASS

Detlef Nauck and Rudolf Kruse
University of Magdeburg
Faculty of Computer Science
Institute of Information and Communication Systems
Universitaetsplatz 2, D–39106 Magdeburg, Germany
Phone: +49.391.67.12700, Fax: +49.391.67.12018
E-Mail: nauck@iik.cs.uni-magdeburg.de, URL: fuzzy.cs.uni-magdeburg.de/∼nauck

Keywords: fuzzy classification, learning, neuro–fuzzy system

Abstract

Neuro–fuzzy classification systems offer means to obtain fuzzy classification rules by a learning algorithm. It is usually no problem to find a suitable fuzzy classifier by learning from data, however, it can be hard to obtain a classifier that can be interpreted conveniently. In this paper we discuss extensions to the learning algorithms of NEFCLASS, a neuro–fuzzy approach for data analysis that we have presented before. We show how interactive strategies for pruning rules and variables from a trained classifier can enhance its interpretability.

1 Introduction

NEFCLASS is used to derive fuzzy rules from a set of data that can be separated in different crisp classes. The fuzzy rules describing the data are of the form:
R : **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 (**NE**uro **F**uzzy **CLASS**ification) is to discover these rules and to learn the shape of the membership functions 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 $R_1, \ldots, R_k$.

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 [3] or hyper–rectangular clusters [1, 2, 8]. Fuzzy sets are then constructed by projecting the clusters. In the case of hyper–ellipsoidal clusters the projection causes a loss of information which is avoided, when hyper–rectangular clusters are used. However, in both cases the induced fuzzy rules are sometimes not easy to interpret. NEFCLASS modifies the fuzzy sets directly, and can therefore use constraints to make sure that the induced fuzzy rules can be interpreted well.

2 The NEFCLASS model

A NEFCLASS system has a 3–layer feedforward architecture that is derived from a generic fuzzy perceptron [5] (see also the accompanying paper at this conference [6]). The first layer U_1 contains the input units representing the pattern features. The activation a_x of a unit $x \in U_1$ is usually equal to its external input. However, it can be different, if the

This paper appeared in Proc. Seventh International Fuzzy Systems Association World Congress IFSA'97, Vol. IV, pp. 50–55, Academia Prague, 1997.

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 activation of a rule unit $R \in U_2$, and the activation of an output unit $c \in U_3$ when a pattern $\mathbf{p}$ is propagated are computed by

$$a_R^{(p)} = \min_{x \in U_1} \{W(x,R)(a_x^{(p)})\},$$

$$a_c^{(p)} = \sum_{R \in U_2} \frac{W(R,c) \cdot a_R^{(p)}}{W(R,c)},$$

$$\text{or alternatively:} \quad a_c^{(p)} = \max_{R \in U_2} \{a_R^{(p)}\},$$

where $W(x,R)$ is the *fuzzy weight* on the connection from input unit x to rule unit R. $W(R,c)$ is the weight on the connection from rule unit R to output unit c. For semantical reasons all these weights are fixed at 1 [4]. Alternatively, the output activation 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 $\varphi(\mathbf{x}) = (c_1, \ldots, c_m)$ 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 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.

NEFCASS' 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. 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.

3 Learning in NEFCLASS

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 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 1 (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 set $\tilde{\mathcal{L}} = \{(\mathbf{p}_1, \mathbf{t}_1), \ldots, (\mathbf{p}_s, \mathbf{t}_s)\}$ 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

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

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

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

than 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 $\left\lfloor \frac{k}{m} \right\rfloor$ 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 $(\mathbf{p}, \mathbf{t})$ 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 $\delta_{c_i} = t_i - a_{c_i}$

(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 $\sigma > 0$:

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

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).

A discussion and an interpretation of the learning procedure of NEFCLASS can be found in [7] and in the accompanying paper at this conference in [6].

4 Pruning a Rule Base

The learning algorithm of NEFCLASS provides good results for many classification problems. An example is given in the accompanying paper at this conference [6]. However, a good interpretation of the learning result cannot always be guaranteed, especially for high-dimensional problems. Because interpretation is one reason for using a fuzzy classifier in the first place, there is a need to enhance the learning algorithms of a neuro-fuzzy system with techniques for simplifying the obtained result. In the following we will use the "Wisconsin Breast Cancer" (WBC) data set [9] (683 cases: 342 for training, 341 for testing) to illustrate the improvements we have implemented in the NEFCLASS learning procedures.

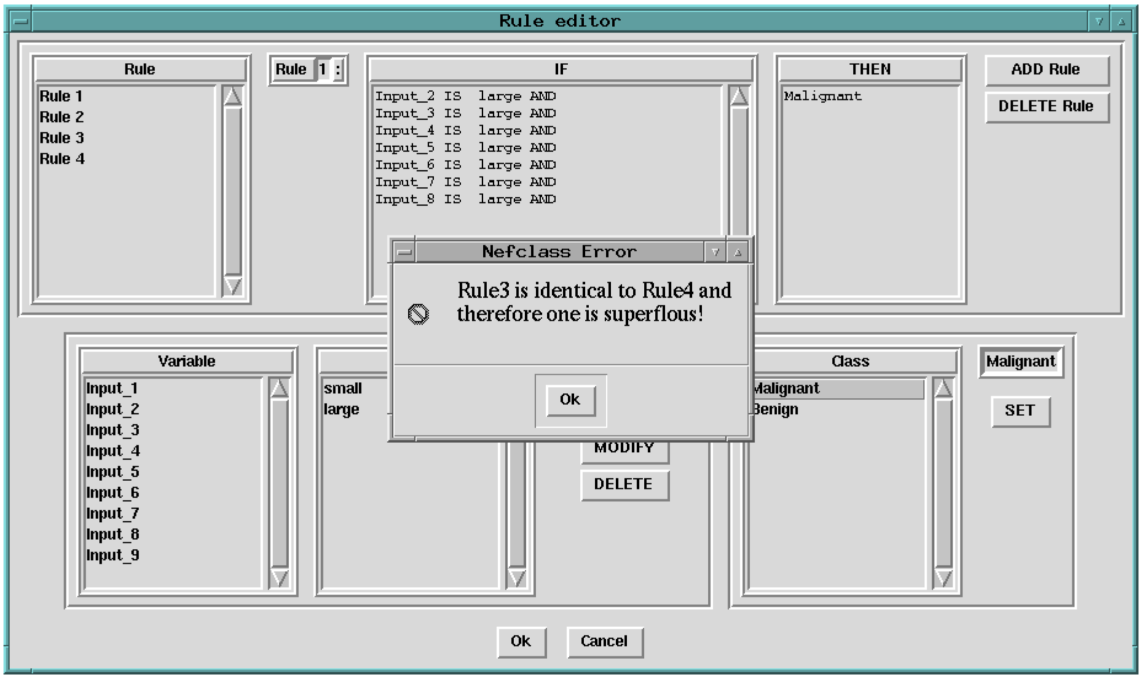

Figure 1: The rule editor of NEFCLASS checks the rule base for consistency

The best result that we obtained with the previous version of NEFCLASS applied to the WBC data yields the four rules given in Table 1. We use "best per class" rule learning, allowed NEFCLASS to create four rules (two for each class), and applied the constraint that the fuzzy sets must intersect at membership degree 0.5. With this training situation we obtained 26 classification errors on the whole data set (8 on the training set, and 18 on the test set, in the following denoted by (8,18)).

Table 1: Rules for the WBC data prior to pruning

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

With the new pruning techniques of NEFCLASS this rule base can be further improved. The rule editor of NEFCLASS supports the user with the rule simplification process. It checks the rule base for consistency after each pruning step, and tells the user which inconsistencies have occured (see Fig. 1). The user can decide what actions have to be taken in case of an inconsistency, for example which rule to delete, if there are two rules with identical antecedents, but different conclusions.

NEFCLASS offers five pruning strategies:

- Evaluate rules: The rule learning procedure of NEFCLASS is invoked again, and only the best k rules are kept. The user can specify a value for k. This strategy is useful, if the learning process was started with a large number of rules, and a many of them have a poor performance measure.

- Delete inputs: NEFCLASS computes the correlations of the input variables with the class information, and offers the possibility to delete input variables that have a low correlation to the output. The user can specify the absolute value for the minimum correlation. The deletion of inputs can lead to inconsistencies in the rule base. The rule editor will notify the user of this and support her or him in solving such problems.

- Delete rules: Rules that are only responsi-

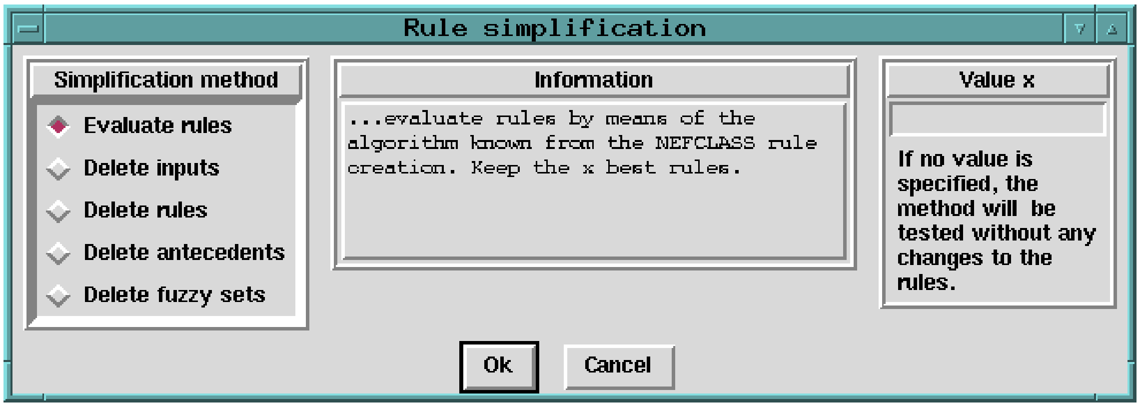

Figure 2: Dialog window for simplifying a NEFCLASS rule base

ble for a few number of classifications can be deleted from the rule base, if they are not needed to cover assumptions in the data (see Fig. 3). The user can specify a percentage value that has to be reached by each rule to remain in the rule base.

- Delete antecedents: For each rule the antecedents that are not identical to the rule's degree of fulfillment in at least z percent, are deleted from the rule. The user can specify a value for z. This method can lead to an inconsistent rule base, which has to be repaired in the rule editor.

- Delete fuzzy sets: If a variable is partitioned by more than two fuzzy sets, sometimes the support for one or more of them can become quite large during learning. This can be seen as evidence, that such a fuzzy set is superfluous. The user can specify a percentage value, and all fuzzy sets that cover more percent of the domain are deleted, which leads to a reduction of variables in the antecedents. This procedure can also lead to an inconsistent rule base, which will be noticed by the rule editor.

Each of the five above mentioned pruning strategies is applied interactively by the user. It is possible to view the evaluations of the pruning strategy which shall be applied prior to using it. By this the user can decide on the appropriate parameter for the pruning method. After the pruning method was applied the rule editor is invoked and displays

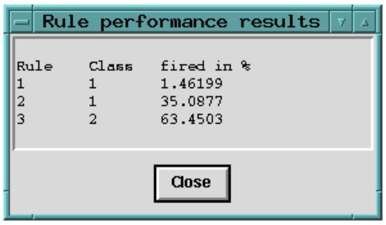

Figure 3: Rules that rarely fire are candidates for deletion

the new rule base. If pruning resulted in inconsistencies or other problems like identical rules, the user can solve these problems step by step. If the new rule base is not desirable, the pruning step can be cancelled. If the classification performance decreases after a pruning step, the learning process can be invoked to improve it again, if possible. By saving and restoring NEFCLASS systems the user can try different strategies to improve the rule base until a set of rules is found that yields acceptable performance and interpretability.

In the following we describe the results on trying to improve the rule base for the WBC data. We begin with the rule base shown in Table 1 which is obtained by learning [7]. We begin the pruning process with deletion of inputs that have a correlation to the output of less than ± 0.7, which leads to the deletion of inputs 1 and 9. As you can see now rules 3 and 4 are identical (see Table 1), and therefore rule 4 is removed. The classification performance is (8,18) like it was before. In the next step we deleted rules that yield the highest activation compared to the other rules in less than 2% of all cases. This leads to deletion of rule 1, and does not change the performance. Both remaining rules now used v_5 *is small* as antecedent, and therefore we deleted this variable from both rules.

As a last step we delete antecedents that are not identical to the rule's degrees of fulfillment in at least z percent. We tried first $z = 7\%$ and $z = 6\%$, but found that the performance decreased and could not be increased again by restarting the learning process. Finally we tried $z = 5\%$ which led to deletion of variable 2 in both rules, and reduced the performance slightly to (10,22). However after training the NEFCLASS system for another 300 epochs, we obtained a classification result of (6,20). So we have again 26 classification errors, but now a much shorter rule base. The two remaining rules can be interpreted as follows:

R_1: If the variables 3,4,6,7 and 8 are all *large*, then the class is *malignant*,

R_2: If the variables 3,4,6,7 and 8 are all *small*, then the class is *benign*.

The linguistic terms *small* and *large* for each variable are represented by membership functions which overlap with their neighbors at membership degree 0.5, and so also the fuzzy sets are nicely interpretable.

5 Conclusions

We have presented new learning strategies for our neuro-fuzzy model NEFCLASS, which allow to simplify a rule base by several interactive pruning methods. The example that we discussed above shows that a neuro-fuzzy system must be used as a development tool. It supports the user to find a desired fuzzy system based on training data, but it cannot do all the work. A good **and** interpretable fuzzy classifier can usually not simply be found by an automatic learning process. The learning and pruning strategies of NEFCLASS are simple and fast heuristics, and they are not supposed to function as an automatic process to create a fuzzy system. The user must be able to supervise and interpret the learning procedure in all its stages. The NEFCLASS software provides this functionality. The software tool and the data that was used by us to obtain the results presented in this paper, can be found on our WWW server fuzzy.cs.uni-magdeburg.de.

References

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

[2] Shigeo Abe, Ming-Shong Lan, and Ruck Thawonmas. Tuning of a fuzzy classifier derived from data. *Int. J. Approximate Reasoning*, 14(1):1–24, January 1996.

[3] Frank Klawonn and Rudolf Kruse. Constructing a fuzzy controller from data. *Fuzzy Sets and Systems*, 85:177–193, 1997.

[4] 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. 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 461–465. ACM Press, New York, February 1995.

[5] Detlef Nauck and Rudolf Kruse. Designing neuro-fuzzy systems through backpropagation. In Witold Pedrycz, editor, *Fuzzy Modelling: Paradigms and Practice*, pages 203–228. Kluwer, Boston, 1996.

[6] Detlef Nauck and Rudolf Kruse. What are neuro-fuzzy classifiers? In *Proc. Seventh International Fuzzy Systems Association World Congress IFSA'97*, volume III, pages 228–233, Prague, 1997.

[7] Detlef Nauck, Ulrike Nauck, and Rudolf Kruse. Generating classification rules with the neuro-fuzzy system NEFCLASS. In *Proc. Biennial Conference of the North American Fuzzy Information Processing Society NAFIPS'96*, Berkeley, CA, June 1996.

[8] 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. 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 466–470. ACM Press, New York, February 1995.

[9] W.H. Wolberg and O.L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. National Academy of Sciences*, 87:9193–9196, December 1990.