Fuzzy Classification Rules using Categorical and Metric Variables

Detlef Nauck and Rudolf Kruse
University of Magdeburg, Faculty of Computer Science
Universitaetsplatz 2, D-39106 Magdeburg, Germany
Tel: +49.391.67.12700, Fax: +49.391.67.12018
E-Mail: nauck@fuzzy.cs.uni-magdeburg.de, WWW: fuzzy.cs.uni-magdeburg.de

Abstract

In real world data sets we often have to deal with different kinds of variables. The data can be for example categorical or metric. Data mining methods can often deal with only one kind of data. Even fuzzy systems that are not dependent on the scales of variables usually only use metric data. In this paper we propose a learning algorithm that creates mixed fuzzy rules – fuzzy rules that use categorical and metric variables.

Keywords: categorical data, data mining, learning, mixed fuzzy rule, neuro-fuzzy system

1 Introduction

When real world data sets must be analysed we often have to deal with different types of variables on different scales, i.e. nominal scales (categorical data), ordinal scales, or interval and ratio scales (both metric). Many approaches that are used in data mining [2] can only deal with one kind of data. For example, neural networks, many statistical procedures and pattern analysis rely on metric data. To process nominal scaled data these approaches represent them usually on artificial metric scales. Approaches like decision trees [13], Bayesian Networks [4, 12] or logic-based approaches work best with categorical data, or at least discrete finite domains. To deal with continuous variables, they must use intervals. Fuzzy systems are not dependent on the scales of the data they process. However, fuzzy system software environments usually assume that variables are metric.

In data mining fuzzy systems can have the advantage to produce solutions that are interpretable in terms of the involved variables. Fuzzy systems can be created by learning from data using fuzzy cluster analysis [1, 3] or neuro-fuzzy methods [5]. Fuzzy cluster analysis depends on a distance measure and can therefore only use metric data. Neuro-fuzzy methods use different approaches to create rule bases. A popular method is to use a hyperbox-oriented approach, where rules are picked from a regular grid of hyperboxes (= multi-dimensional fuzzy sets) [9, 14].

It would be useful to be able to create fuzzy rules from data that contain categorical variables without converting them to a metric scale. In this paper we call fuzzy rules that contain categorical and metric variables *mixed fuzzy rules*. We propose an algorithm that can create a rule base of mixed fuzzy rules. The algorithm is an extension to our NEFCLASS model (neuro-fuzzy classification) [6, 7].

Mixed fuzzy rules demand to use fuzzy sets for the categorical variables that cannot be represented by the usual parameterized membership functions like triangles or trapezoids. In Section 2 we describe the kind of rules and discuss an artifical example. In Section 3 we present the learning algorithm and in Section 4 we apply our algorithm to real world data set. In Section 5 we discuss the interpretation of mixed fuzzy rules and conclude with an outlook on future work on the algorithm.

2 Using Categorical Variables in Fuzzy Rules

Let us consider two attributes x and y, where $x \in X \subseteq \mathbb{R}$ is continuous and $y \in Y = \{A, B, C\}$ is categorical. In a fuzzy rule we describe values of x by linguistic terms. We use *lvalue* to denote any such linguistic term (*lvalue* may be a term like *small, approximately zero, large*, etc.). In a mixed fuzzy rule using two variables we can have the following situations:

(i) fuzzy-exact: if x is *lvalue* and y $= A$ then ...

(ii) fuzzy-imprecise: if x is *lvalue* and y $\in \{B, C\}$ then ...

(iii) fuzzy-fuzzy: if x is *lvalue* and y is $\{(A, \mu(A)), (B, \mu(B)), (C, \mu(C))\}$ then ...

In the first two cases the categorical variable y has a "switching function" for a rule. If y does not assume one of the values noted in the respective y-term of the antecedent, the rule is not applicable at all. But if y does assume any of these values, the applicability of the rule is not restricted by this argument, and the degree of fulfilment only depends on the value for x.

In the third situation, we use a fuzzy set to describe the value that y may assume, by simply attaching a degree of membership to each element of Y using some membership function $\mu : Y \to [0, 1]$. By giving some value to y we can now restrict the applicability of the rule to any degree between 0 and 1.

Obviously case (i) and (ii) are just special cases of case (iii), because we can replace $y = A$ by y is $\{(A, 1), (B, 0), (C, 0)\}$ and $y \in \{A, B\}$ by y is $\{(A, 1), (B, 1), (C, 0)\}$.

Because the elements of Y are not ordered, we cannot easily use a linguistic term to label fuzzy sets like $\{(A, \mu(A)), (B, \mu(B)), (C, \mu(C))\}$. This means the interpretability in terms of the variables is restricted compared to fuzzy rules that just use variables on metric scales. We will discuss this issue in Section 5.

For cases (i) and (ii) we can remove the categorical variable from the fuzzy rules, and create different rule bases, one for each combination of values of y (see Fig. 1). Depending on the value of y we simply select the applicable rule base. Such a situation may be, for example,

useful in a medical setting, where we want to classify diseases of patients according to some symptoms. If we assume that the classification depends on whether the patient is female or male, we can simply build two different rule bases – one to classify female patients, and one to classify male patients.

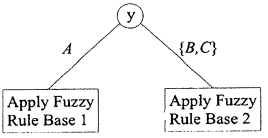

Figure 1: Switching between fuzzy rule bases depending on a categorical variable

For case (iii) let us consider an artifical data set given in Fig. 2. For x we want to use three fuzzy sets labelled by *small, medium* and *large*. The degree of membership for any value adds up to 1.0, i.e. the membership functions overlap at degree 0.5. The vertical lines visualise these points. The values of y are represented at the vertical scale, but please note that the scale is nominal, i.e. the ordering of elements has no meaning.

We assume that the data can be classified into a *positive* class and a *negative* class, as denoted by the + and − signs in the data space. As can be seen in Fig. 2 there is some degree of overlapping between both classes, especially in the area where x is *small*.

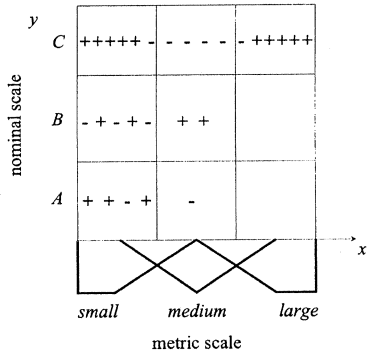

Figure 2: An artificial data set with a metric and a categorical attribute

What kind of rules can we use to represent the classification of this data set? If we look at the grid drawn in Fig. 2, we could get the idea to use 7 rules, one for each box that contains data, and label each rule with the majority class. This is what would probably happen, if we apply some kind of clustering algorithm by ignoring the case that y is nominal scaled and choose to map it to a metric scale, e.g. by setting $A = 1$, $B = 2$, $C = 3$.

However, we do not want to hide the fact that y is categorical and choose another approach. Let us first consider the situation, where x is *small* and the class is *positive*. We count 3 cases where $y = A$, 2 cases where $y = B$ and 5 cases where $y = C$. By normalising these values, we obtain the fuzzy set $\{(A, 0.6), (B, 0.4), (C, 1.0)\}$. If we consider the *negative* class and x is *small*, we obtain the fuzzy set $\{(A, 0.33), (B, 1.0), (C, 0.33)\}$ in the same manner.

This means, we can add the following two rules to our rule base:
R_1: if x is *small* and y is $\nu_1 = \{(A, 0.6), (B, 0.4), (C, 1.0)\}$, then class is *positive*
R_2: if x is *small* and y is $\nu_2 = \{(A, 0.33), (B, 1.0), (C, 0.33)\}$, then class is *negative*

Let us now assume that we obtain three new cases where $x = x_0$ in each case with $\mu_{small}(x_0) = 1.0$ and y having a different value in each case. Tab. 1 gives the degrees of fulfilment for both rules and the classification for all three cases.

Table 1: Classification of three sample cases using rules R_1 and R_2

x	$\mu_{small}(x)$	y	$\nu_1(y)$	$\nu_2(y)$	R_1	R_2	class
x_0	1.0	A	0.6	0.33	0.6	0.33	*positive*
x_0	1.0	B	0.4	1.00	0.4	1.00	*negative*
x_0	1.0	C	1.0	0.33	1.0	0.33	*positive*

The rules are, like rules that use only metric variables, representations of typical representatives for the classes. If we consider the positive class, a typical case would for example have a *small* value for x and either C, A or B for y, where we would consider C more typical than A and A more typical than B. For the *negative* class we would consider B more typical than A or C.

R_1 and R_2 are partially contradictory, as they overlap in all variables. If we use

$$\theta_x(\mu_1, \mu_2) = \sup_x \min\{\mu_1(x), \mu_2(x)\}$$

to denote the degree of similarity or overlapping of two fuzzy sets μ_1 and μ_2, then we obtain $\theta = 0.4$ for the antecedents of rules R_1 and R_2. Rules with $\theta = 0$ are mutually exclusive, and for $\theta = 1$ the rules are either identical or one rule is a generalisation of the other one (if the consequents are identical), or they are completely contradictory (if the consequents are different). Partial contradiction is, however, common for fuzzy rule bases, as overlapping of rules is a desired feature.

If we use this rule generation technique further, we obtain the following two rules next:
R_3: if $x$is *medium* and y is $\nu_3 = \{(A, 0.25), (B, 0), (C, 1.0)\}$, then class is *negative*,
R_4: if x is *medium* and y is $\nu_4 = \{(A, 0), (B, 1.0), (C, 0)\}$, then class is *positive*.

For R_3 and R_4 we obtain $\theta = 0$, i.e. they are mutually exclusive. In this case it would be possible to use crisp sets to describe the values for y ($y \in \{A, C\}$ for R_3 and $y = B$ for R_4). However, we would loose the information that the combination x is *medium* and $y = C$ is much more typical for the *negative* class then the combination x is *medium* and $y = A$. It is therefore useful to keep the fuzzy set representation.

For the last box in Fig. 2 that contains data, we obtain a problem. Using the procedure described above, we obtain the two contradictory rules ($\theta = 1$):
R_5: if $x$is *large* and y is $\nu_5 = \{(A, 0), (B, 0), (C, 1.0)\}$, then class is *positive*,
R_6: if x is *large* and y is $\nu_6 = \{(A, 0), (B, 0), (C, 1.0)\}$, then class is *negative*.

We cannot include both rules in our rule base, as we only tolerate partial contradiction. Therefore we delete the less performing rule, i.e. the rule that would cause most misclassifications: in this case rule R_6. Our final rule base consists therefore of rules $R_1, \ldots, R_5$.

3 Learning Mixed Fuzzy Rules

The algorithm given in Fig. 3 computes a fuzzy rule base from a set of training data containing categorical and metric data.

We use the following notations: $R = (\mathbf{a}, \mathbf{M}, c)$ is a fuzzy rule. $\mathbf{a}$ is vector of fuzzy sets and part of the antecedent of a rule, μ is a membership function over a metric variable in an antecedent. $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_s]$ is an array of vectors, where the vector $\mathbf{m}_i$ represents a fuzzy set for the ith categorical variable. $\mathbf{a}$ and $\mathbf{M}$ together represent the antecedent of a fuzzy rule. c represents a class index, i.e. a consequent of a rule. When a fuzzy rule is created, $\mathbf{M}$ is initialized (Fig. 3, line 8) such that each $\mathbf{m}_i$ contains only zeros. The training data set L contains pairs $(\mathbf{p}, \mathbf{t})$, where $\mathbf{p}$ is an input pattern consisting of metric and categorical featuers, and $\mathbf{t}$ is target vecotor describing the class of $\mathbf{p}$. We expect, that there are initial fuzzy sets for each metric variable.

At first the algorithm creates a set of rule base candidates. This set may be inconsistent, as it contains contradictory rules. After resolving inconsistencies, by selecting from multiple rules with identical antecedents but different consequents the best performing rule, a final rule base is created. This is done, by applying one of the rule evaluation algorithms of NEFCLASS [8].

After rule creation the fuzzy sets are trained to improve the performance of the classifier. Membership functions of metric variables are trained according to the algorithms given in [5]. To compute the modifications for a fuzzy set $\mathbf{m}_j^{(r)}$ of a categorical variable x_j that is used in the antecedent of rule r the following procedure is used:

$$
\begin{aligned}
\Delta\mathbf{m}_j^{(r)}[p_j] &= m \cdot e_r \\
m &= \begin{cases} \mathbf{m}_r[p_j] & \text{if } e_r < 0 \\ 1 - \mathbf{m}_r[p_j] & \text{if } e_r > 0 \end{cases} \\
e_r &= e_{c_r} \cdot (\tau_r \cdot (1 - \tau_r) + \varepsilon) \\
e_{c_r} &= t_{c_r} - o_{c_r}
\end{aligned}
$$

where p_j is the value for x_j in the current input pattern $\mathbf{p}$, t_{c_r} is the target value for class c_r that is used as the consequent of rule r, o_{c_r} is the output of the classifier for class c_r and τ_r

```
01:  /* find all rectangular fuzzy clusters that contain data */
02:  for each pattern (p, t) of L do
03:     begin
04:        for each metric input feature p_i do
05:           find μ_{j_i}^{(i)} such that μ_{j_i}^{(i)}(p_i) = max_{j∈{1,...,q_i}} {μ_j^{(i)}(p_i)};
06:        Create the first part of the antecedent a = (μ_{j_1}^{(1)}, ..., μ_{j_n}^{(n)});
07:        c = class index of p given by t;
08:        initialize M;
09:        Create rule R = (a, M, c);
10:        If R is not in list of rule base candidates
11:           then add R to list of rule base candidates;
12:     end;
13:
14:  /* for each rule base candidate, determine frequencies for categorical variables */
15:  create a new empty list of rule base candidates
16:  for each pattern (p, t) of L do
17:     for each rule base candidate R do begin
18:        copy rule base candidate R to R'
19:        for each class c do
20:           for each categorical feature p_i do
21:              if class of p = c
22:                 then with rule candidate R': m_i[p_i] = m_i[p_i] + 1;
23:              add R' to the new list of rule base candidates
24:     end
25:  delete the old list of rule base candidates;
26:
27:  /* transform the m_i into fuzzy sets by normalizing them */
28:  for each rule base candidate R do
29:     normalize all m_i;
30:
31:  /* resolve conflicts and select final rule base */
32:  Find all contradicting rules and resolve conflicts;
33:  Select a rule base from the rule base candidates using a performance measure;
```

Figure 3: Algorithm for creating a rule base of mixed fuzzy rules from data

is the degree of fulfilment of rule r. The modifications are computed after each presented training pattern and are applied to the fuzzy set after all patterns were processed (offline learning, batch learning). The rule error e_r is largest for degree of fulfilment $\tau = 0.5$. The goal is to drive the membership degrees to the extreme values during learning. For that we use a small value ε to make sure, that fuzzy sets that yield a degree of membership of 1.0 or 0.0 for some value can still be modified by the learning process.

The algorithm is implemented in NEFCLASS-J the new implemention of the NEFCLASS model. The software is written in Java and will be available for free via the Internet. In the next section we describe the application of the algorithm using some real world data.

4 Application of the Learning Algorithm

For testing the algorithm described in the previous section, we used the "Wisconsin Breast Cancer" data set (WBC data) [15]. We used our new software tool NEFCLASS-J (see Fig. 4) that provides all the learning algorithms of our NEFCLASS model that were described elsewhere [7, 10] and some additional features [11] including the algorithm to handle categorical variables.

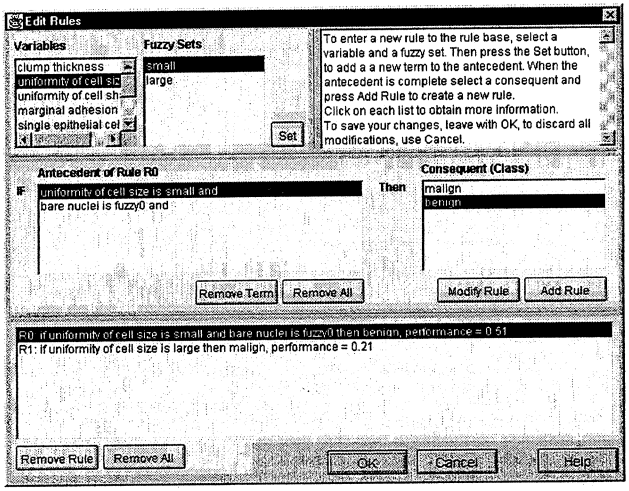

Figure 4: The rule editor of NEFCLASS-J displaying the learning result

The WBC data set is available via the Internet from the machine learning repository (ftp://ftp.ics.uci.edu/pub/machine-learning-databases). It is a breast cancer database that was provided by W.H. Wolberg from the University of Wisconsin Hospitals, Madison [15]. The data set contains 699 cases and 16 of these cases have missing values. Although the NEFCLASS-J can handle data with missing values [11], we deleted those cases, because other approaches we compare our result to (see Tab. 3) cannot handle missing values.

Each case is represented by an identification number and nine attributes (x_1: clump thickness, x_2: uniformity of cell size, x_3: uniformity of cell shape, x_4: marginal adhesion, x_5: single epithelial cell size, x_6: bare nuclei, x_7: bland chromatin, x_8: normal nucleoli, x_9: mitoses). All attributes are from the domain $\{1, \ldots, 10\}$. Each case belongs to one of two classes (benign: 458 cases, or malignant: 241 cases). The values of all nine variables are actually from an ordinal scale. Classifier usually simply treat them as metric values and good classification results can be obtained this way (see Tab. 3).

To test our new learning algorithm we chose to interpret variables x_3 and x_6 as categorical

variables and the rest as metric variables. We chose x_3 and x_6 because these two variables are usually considered as influential by other classification approaches applied to the WBC data.

The new NEFCLASS-J tool offers an automatic creation and pruning process that creates a rule base, trains the membership functions and finally automatically prunes the classifier to remove as many variables and rules as possible. This process can be combined with an n-fold cross validation. For this, the tool randomly splits the data set into n parts and in each cycle it uses $n-1$ parts for creating the classifier and the remaining part for validation such that each part is once used for validation. The final classifier is obtained by using all available data. The error estimation (mean error) $\bar{e}$ for unseen data is computed from the errors e_i of the classifiers created during validation.

The 99% confidence interval for the estimated error is given by

$$\bar{e} \pm 2.58 \cdot \hat{\sigma}_{\bar{e}}$$

where $\hat{\sigma}_{\bar{e}}$ is the standard error of the mean:

$$\hat{\sigma}_{\bar{e}} = \sqrt{\frac{\sum_{i=1}^{n}(e_i - \bar{e})^2}{n \cdot (n-1)}}.$$

We used a 10-fold cross validation, and let the tool select the best two rules per class during rule learning. For each metric variable two initial membership functions were given (half trapezoids, compare Fig. 5). The fuzzy sets for the categorical variables were created during rule learning. The fuzzy sets were trained until the error on the validation set could not be further decreased, but not longer than 300 cycles.

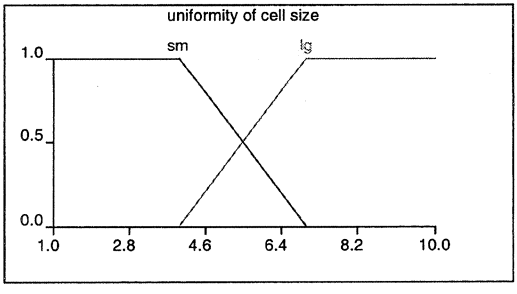

Figure 5: Initial membership functions for metric variables

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

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

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

The membership functions after training are shown in Fig. 6. The fuzzy set for the categorical variable x_6 is drawn as a histogram. Its exact representation is

$$term_1^{(6)} = \{(1, 1.0), (2, 0.99), (3, 0.75), (4, 0.63), (5, 0.68), (6, 0.0), (7, 0.14), (8, 0.02), (9, 0.0), (10, 0.22)\}.$$

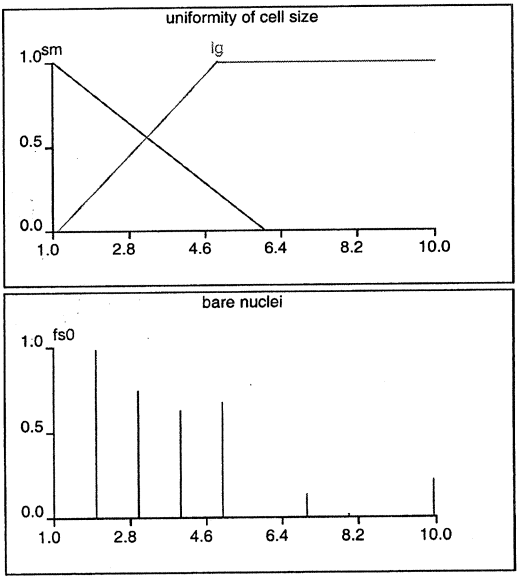

Figure 6: Membership functions for the metric variable x_2 and the categorical variable x_6 after training

This classifier causes 29 misclassifications (4.25%) on the *training data*, i.e. its classification rate is 95.75% (see Tab. 2). The error estimation for *unseen data* obtained from cross validation yields $3.95\% \pm 1.88\%$ misclassifications, i.e. an estimated classification rate of $96.05\% \pm 1.88\%$ (99% confidence interval). This error estimation must be interepreted this way: A classifier that is obtained by the described learning procedure and using the described parameters and training data is estimated to produce an error of $3.95\% \pm 1.88\%$ on unseen data.

The final rule base was also discovered in three of the validation cycles. Altogether seven different rule bases were discovered during validation (nine rule bases with two rules, one rule base with three rules). However, most of the other rule bases were very similar and differed only in additionally using the other categorical variable x_3, using x_3 instead of x_2, or using just x_2.

Table 2: The confusion matrix of the final classifier obtained by NEFCLASS-J

	Predicted Class							
	malign		benign		not classified		sum	
malign	222	(32.50%)	17	(2.49%)	0	(0.00%)	239	(34.99%)
benign	12	(1.76%)	432	(63.25%)	0	(0.00%)	444	(65.01%)
sum	234	(34.26%)	449	(65.74%)	0	(0.00%)	683	(100.00%)

correct: 654 (95.75%), misclassified: 29 (4.25%), error: 58.32.

Tab. 3 compares the result obtained with NEFCLASS-J (last entry) to results obtained with other approaches including a previous release of NEFCLASS for Unix workstations (NEFCLASS-X [10]) The classification performance on unseen data is very good and the classifier is very compact. The error estimates given column "Validation" of Tab. 3 are either obtained from 1-leave-out cross validation, 10-fold cross validation, or from testing the solution once by holding out 50% of the data for a test set.

Table 3: Comparing the NEFCLASS learning outcome for the WBC data set to some other approaches. Numbers in () are mean values from cross validation. The column "Error" contains an estimated error for unseen data

Model	Tool	Remarks	Error	Validation
Discriminant Analysis	SPSS	linear model 9 variables	3.95%	1-leave-out
Multilayer Perceptron	SNNS	4 hidden units, RProp	5.18%	50% test set
Decision Tree	C4.5	31 (24.4) nodes, pruned	4.9%	10-fold
Rules from Decision Tree	C4.5rules	8 (7.5) rules using 1–3 variables	4.6%	10-fold
NEFCLASS (metric variables)	NEFCLASS-X (Unix version)	2 (2.1) rules using 5–6 variables	4.94%	10-fold
NEFCLASS (2 categorical variables)	NEFCLASS-J (Java version)	2 (2.1) rules using 1–3 variables	3.95%	10-fold

5 Conclusions

We have presented a learning algorithm for mixed fuzzy rules using training data with categorical and metric variables. The resulting fuzzy rules cannot be as easily interpreted as fuzzy rules that use only metric variables and continuous membership functions that can be labelled with terms like *small* or *large*.

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

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

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

The algorithm presented in this paper is an extension to our neuro-fuzzy model NEFCLASS. The new Java version of the NEFCLASS software will be publicly available at the time of this conference at http://fuzzy.cs.uni-magdeburg.de. The other new features of this software will be described in another paper [11].

Literature

[1] James C. Bezdek and Sankar K. Pal, editors. *Fuzzy Models for Pattern Recognition.* IEEE Press, New York, 1992.

[2] M. Fayyad, Usama, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* MIT Press, Menlo Park, CA, 1996.

[3] Frank Höppner, Frank Klawonn, Rudolf Kruse, and Thomas Runkler. *Fuzzy Cluster Analysis.* Wiley, Chichester, 1998. to appear.

[4] Rudolf Kruse, Erhard Schwecke, and Jochen Heinsohn. *Uncertainty and Vagueness in Knowledge-Based Systems: Numerical Methods.* Springer-Verlag, Berlin, 1991.

[5] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Foundations of Neuro-Fuzzy Systems.* Wiley, Chichester, 1997.

[6] Detlef Nauck and Rudolf Kruse. A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems*, 89:277–288, 1997.

[7] Detlef Nauck and Rudolf Kruse. NEFCLASS-X – a soft computing tool to build readable fuzzy classifiers. *BT Technology Journal*, 16(3):180–190, 1998.

[8] Detlef Nauck and Rudolf Kruse. Neuro-fuzzy methods in fuzzy rule generation. In Didier Dudois and Henri Prade, editors, *Approximate Reasoning and Fuzzy Information Systems*, Handbook of Fuzzy Sets, chapter 5. Kluwer, Norwell, MA, 1998.

[9] Detlef Nauck and Rudolf Kruse. Neuro-fuzzy systems. In E. Ruspini, P. Bonissone, and W. Pedrycz, editors, *Handbook of Fuzzy Computation*, chapter D.2. Institute of Physics Publishing Ltd., Philadelphia, PA, 1998.

[10] Detlef Nauck and Rudolf Kruse. Obtaining interpretable fuzzy classification rules from medical data. *Artificial Intelligence in Medicine*, 1999. Accepted.

[11] Detlef Nauck, Ulrike Nauck, and Rudolf Kruse. NEFCLASS for JAVA – new learning algorithms. 1999. Submitted.

[12] J. Pearl. *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, 1988.

[13] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA, 1993.

[14] L.-X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. Syst., Man, Cybern.*, 22(6):1414–1427, 1992.

[15] 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.