PERGAMON

Expert Systems with Applications 15 (1998) 375–381

Expert Systems with Applications

An initial comparison of a fuzzy neural classifier and a decision tree based classifier

Jurgen Martens*, Geert Wets, Jan Vanthienen, Christophe Mues

Department of Economics and Applied Economics, Catholic University of Leuven, Naamsestraat 69, 3000 Leuven, Belgium

Abstract

At the present time a large number of AI methods have been developed in the field of pattern classification. In this paper, we will compare the performance of a well-known algorithm in machine learning (C4.5) with a recently proposed algorithm in the fuzzy set community (NEFCLASS). We will compare the algorithms both on the accuracy attained and on the size of the induced rule base. Additionally, we will investigate how the selected algorithms perform after they have been pre-processed by discretization and feature selection. © 1998 Elsevier Science Ltd. All rights reserved

1. Introduction

Pattern classification problems (Kuncheva, 1996), consist of assigning to an object, described as a point in a certain feature space S^n, a class label ω_i from a predefined set $\Omega = \{\omega_1,\omega_2,\ldots,\omega_M\}$. The problem of designing a classifier is to find a mapping $D{:}S^n \to \Omega$, optimal in the sense of a certain criterion $J(D)$. It is a well-known result in statistical decision theory that the optimal classifier in terms of J (called also the Bayesian classifier) is the one that assigns to an arbitrary n-tuple in S^n the class label ω^* corresponding to the highest posterior probability, i.e. $\omega^* = \arg\max_\omega (P(\omega|\mathbf{x}^{(n)})$. In contrast to statistical classification procedures, the main goal in recently developed AI classification techniques is not the construction of an (asymptotically) Bayesian optimal classifier, but the preclusion of the fact that the entire classification process becomes a black box phenomenon. The AI classifier can then be interpreted as a mapping $\tilde{D}$, which contains a collection of ‘‘if…then’’ rules of the following global structure:

$$R_k \equiv (\textbf{if } (x_1^{(n)} \epsilon A_{1,k}) \textbf{ and } (x_2^{(n)} \epsilon A_{2,k}) \textbf{and} \ldots \textbf{and} (x_n^{(n)} \epsilon A_{n,k})$$

$$\textbf{then} \text{ the pattern } \mathbf{x^{(n)}} \text{ belongs to class } \omega_l$$

where the terminology $x_j^{(n)}$ is used to denote the jth component or feature in an n-tuple $\mathbf{x}^{(n)}$ in S^n. The terms $A_{1,k}\ldots A_{n,k}$ represent sets of elements whose particular values, together with the label ω_l have to be determined by the applied AI method. At the present time there exists a wide variety of miscellaneous AI algorithms that have been developed in the context of many research projects in the field of classification. Besides classical classification approaches, currently approaches are emerging which are based on fuzzy set theory (Zadeh, 1965). This theory allows one to deal with vague concepts. Fuzzy statistical classification techniques combine essentially the insights in fuzzy set theory with existing classification procedures. An excellent introduction to the domain of fuzzy classifiers can be found in Bezdek (1981).

In the present paper, we will compare a well-known classification technique in the field of machine learning, i.e. C4.5 (Quinlan, 1993) and compare its performance on some benchmarking datasets with a fuzzy classification algorithm, i.e. NEFCLASS (Nauck et al., 1996). Besides comparing the performance on the original datasets we will also investigate how pre-processing of the data by means of discretization and feature selection will influence the results.

The organization of the paper is as follows. In the first and the second section, the classification algorithms (C4.5 and NEFCLASS) used in the experiments are described. Next, the performance of both algorithms is compared using seven datasets. We compared the performance both before and after discretization and feature selection. Finally, some concluding remarks are given.

2. C4.5

The C4.5 machine learning program was originally written by Quinlan in the 1980s and has undergone lots of

* Corresponding author. Tel.: +0032-16-326888; E-mail: Jurgen.Martens@econ.kuleuven.ac.be

0957–4174/98/$ - see front matter © 1998 Elsevier Science Ltd. All rights reserved.
PII: S0957-4174(98)00057-8

major modifications since then. The essence of the algorithm boils down to the construction of a classification tree where every level coincides with a particular feature in the feature space S^n and the branches correspond to certain attribute values or ranges. What follows is only a compact overview of the algorithm. A comprehensive treatment is given in Quinlan (1993).

C4.5 has been called a *greedy* algorithm since it applies some kind of steepest descent heuristic to optimize the structure of the classification tree. Basically, the main goal of the heuristic consists of the construction of a tree where every leaf node contains only elements $\mathbf{x}^{(n)}$ with the same class label ω_l by exploiting some sort of recursive partitioning method of the feature space. In that respect, the algorithm makes use of the information based *gain* and *gain ratio* criteria to effectively construct a tree-like structure with high classification accuracy. In order to determine which feature and which feature values are best suited to be assigned to a certain node and its corresponding branches, C4.5 calculates the global entropy reduction of the dataset by considering all possible scenarios of associating a particular attribute to the node at issue and partitioning the entire feature range into miscellaneous subsets (branches). The entropy decline (gain) at the root level is then given by means of the following expression:

$$-\sum_{j=1}^{k}\frac{\mathrm{freq}(C_j,T)}{|T|}\times\log_2\left(\frac{\mathrm{freq}(C_j,T)}{|T|}\right)-\sum_{i=1}^{n}\frac{|T_i|}{|T|}\times\left[-\sum_{j=1}^{k}\frac{\mathrm{freq}(C_j,T_i)}{|T_i|}\times\log_2\left(\frac{\mathrm{freq}(C_j,T_i)}{|T_i|}\right)\right]$$

where T is the entire dataset, C_j represents the number of n-tuples of class j, n gives the number of branches at the root node and T_i stands for the collection of n-tuples to be positioned in subset i. Once a feature and its corresponding feature subsets are identified at the root level in the classification tree, the algorithm moves on to the next level and again computes the maximum possible downfall in global entropy. Sometimes C4.5 employs the gain ratio criterion instead of the gain measure to cope with the disadvantageous effects of dividing a dataset into a too large a number of subsets at a particular node. The gain ratio of a scenario Φ is hereby defined as:

$$\text{gain ratio}(\Phi)=\text{gain}(\Phi)/\text{split info}(\Phi)$$

$$\text{split info}(\Phi)=\sum_{i=1}^{n}\frac{|T_i|}{|T|}\times\log_2\left(\frac{|T_i|}{|T|}\right)$$

The resulting classification tree one obtains by recursively partitioning a training dataset as described above often gives bad performance when practiced on a set of unseen test cases. The C4.5 algorithm applies an error-based pruning strategy to deal with the inconvenient drawbacks one is confronted with when overtraining or overfitting of classification trees has occurred. As a matter of fact, C4.5 calculates for each classification node a kind of predicted error rate based on the total aggregate of misclassifications at that particular node. The error rate is calculated as the upper limit of an α% confidence interval for the mean E/N of a binomial distribution $B(E/N)$ where E/N is the proportion of misclassifications at the node at issue. The error-based pruning technique essentially boils down to the replacement of vast subtrees in the classification structure by singleton nodes or simple branch collections if these actions contribute to a drop in the overall error rate of the root node.

After the execution of the recursive partitioning algorithm, and the aforementioned error-based pruning, one has obtained a complete classification structure that could directly be applied to classify a wide set of unseen cases with high accuracy. However, the inconvenient representation of the classification process by means of a treelike outline compels the transformation into some kind of rule-based knowledge entity. Therefore, an initial rule base is obtained by top-down progressing the entire classification tree and creating a rule for every single path encountered. The resulting rule base might, however, contain a substantial amount of complex classification rules. Hence, a reduction in rule base magnitude and complexity can be achieved by eliminating certain conditions in a rule premise. C4.5 determines the potential benefits of removing sets of conditions by calculating error estimates for every possible condition reduction scenario. In that respect, error estimates of an arbitrary rule R and its reduced premise part version $R\backslash\{\varepsilon\}$ are given by E_1/N_1 and $(E_1+E_2)/(N_1+N_2)$ where N_1 is the total of n-tuples that satisfy ε, N_2 is the complement of N_1, E_1 is the proportion of N_1 elements that do not belong to the class $\omega_{i(\mathbf{R})}$ and E_2 is the fraction of N_2 cases that do not possess the class label $\omega_{i(\mathbf{R})}$. By analogy with the process of error-based pruning, upper limits are calculated for the above error estimates. A condition set ε is then removed from R if the global error rate of $R\backslash\{\varepsilon\}$ is lower than the one for R. When the total of different condition reduction scenarios to be examined becomes too large, simulated annealing is applied.

3. NEFCLASS

The fuzzy neural network NEFCLASS was been created at the Technical University of Braunschweig, Germany, around 1995. The NEFCLASS learning algorithm comes down to the construction of a three-level multi-layer Perceptron network structure with fuzzy activation functions for units at the intermediate (hidden) level. What follows is a brief introduction of the fuzzy logic reasoning concept, together with a compact overview of the working method of NEFCLASS. Detailed expositions of fuzzy set theory are widely available (e.g. Zimmermann, 1991) while a thorough explanation of NEFCLASS (and some other fuzzy neural networks) can be found in Nauck et al. (1996).

3.1. Fuzzy set theory fundamentals

Fuzzy set theory was originally introduced by Zadeh (1965) in order to deal with non-precise and vague information. In classical set theory, a subset ϑ of X (the universe) can be written as a characteristic function ψ_ϑ, which associates every element in X with a value 0 or 1. In that way elements of X either do or do not belong to ϑ. A fuzzy subset ϑ_{fuz} on the other hand combines every member of X with a value in the continuous interval [0,1]. The characteristic function of ϑ_{fuz} is then given by means of a so-called membership function $\mu_{\vartheta_{\text{fuz}}}$ which depicts some kind of mathematical function on the universe. In the remainder of the text, $\vartheta_{\text{fuz}}(x)$ is replaced by $\vartheta(x)$.

In order to be able to perform classical set theory operations on fuzzy sets, the intersection and union of two fuzzy sets ϑ_1 and ϑ_2 are defined as:

$$(\vartheta_1 \cap \vartheta_2)(x) = \min\{\vartheta_1(x), \vartheta_2(x)\}\ (\vartheta_1 \cup \vartheta_2)(x) = \max\{\vartheta_1(x), \vartheta_2(x)\}$$

The above min and max operators represent the so-called Zadeh t-norms and t-conorms to model the intersection and union operation of fuzzy sets. In addition to this min/max pair of operators, sometimes other designs of t-norms and t-conorms are used (Weber, 1983). The definition of the above intersection and union operators enables us in fact to establish an exhaustive table to depict the truth for some combined propositions of ϑ_1 and ϑ_2. In order to complete this table with truth values for fuzzy implications, the concept of a fuzzy relation has to be introduced. Fuzzy relations are to be considered as fuzzy sets of Ω-tuples and denoted as:

$$\int_{X_1} \cdots \int_{X_\Omega} \mu_{\vartheta_{\text{fuz}}}(x_1, \ldots, x_\Omega)/(x_1, \ldots, x_\Omega)$$

where this integral has to be read as some kind of idempotent integration which identifies for every element $(x_1,\ldots,x_\Omega)$ in the S^Ω space its membership value $\mu_{\vartheta_{\text{fuz}}}$ with respect to the fuzzy relation. A possible relation when $\Omega = 2$ and $X_1 = X_2 = \{1,2,3\}$ might, for example, shade the concept ''is approximately equal to'' and be stated as $1/(1,1) + 1/(2,2) + \ldots + 0.8/(1,2) + \ldots + 0.3/(3,1)$. Fuzzy implications can now generally be written as a relation between a disjunction of t-norms or a conjunction of t-conorms on one side and the fuzzy set in the conclusion part of the implication on the other side. In that respect, a rule of the form ''if x_1 is ϑ_1 and x_2 is ϑ_2 then y is ω'' is jotted down as a relation Γ where $\Gamma = \iota(\Im(\vartheta_1,\vartheta_2)\omega)$; ϑ_1, ϑ_2 and ω are fuzzy subsets in the respective universes X_1, X_2 and Y; ι is an implication operator; and $\Im$ represents a t-norm. The membership function of Γ can then easily be obtained by substituting the fuzzy subsets in the latter expression by their corresponding membership functions.

While the above rationale empowers us in modelling fuzzy implications by means of a fuzzy relation, implication-based reasoning enforces a way of composing fuzzy relations. Zadeh (1979) came up with the theory of approximate reasoning as a framework for dealing with imprecise information and performing basic inference procedures, very much like the classical modus ponens and modus tollens principles. According to Zadeh's theory, the inference of a rule (an implication Γ) ''if x equals ϑ_1 then y equals ϑ_2'' with the fact (Λ) that ''x equals $\vartheta_1^{'}$'' gives rise to the statement (Δ) that ''y equals $\vartheta_2^{'}$'' where $\vartheta_2^{'}$ is defined as $\vartheta_2^{'} = \vartheta_1^{'} \circ \Gamma$. In order to determine the membership function of the fuzzy conclusion set $\vartheta_2^{'}$ one extends $\vartheta_1^{'}$ into the Γ-universe, computes the intersection between Γ and the extension of $\vartheta_1^{'}$ and finally projects the resulting set on the domain of y:

$$\text{proj}_Y(\Gamma \cap \text{cext}_{X\times Y}(\vartheta_1^{'}))$$

If one defines Γ as a relation on X with $X = \times_{i=1}^{\Omega} X_i$, $(i_1,i_2,\ldots,i_k)$ as a subsequence of $(1,2,\ldots,\Omega)$, $(j_1,j_2,\ldots,j_1)$ as the complementary subsequence of $(1,2,\ldots,\Omega)$ and Y as $\times_{m=1}^{k} X_{i_m}$, the projection of Γ on Y is then determined by:

$$\text{proj}_Y(\Gamma) = \int_Y \sup_{x_{j_1},\ldots,x_{j_1}} \mu_\Gamma(x_1, \ldots, x_\Omega)/(x_{i_1}, \ldots, x_{i_k})$$

Carrying out a projection essentially eliminates the universes $X_{j_1}\ldots X_{j_l}$ and associates each vector $(x_{i_1}, \ldots, x_{i_k})$ of the projected fuzzy set with a membership value, equal to the supremum of the original membership function on a domain, set up by the l-dimensional hyperplane $\text{hyp}_{x_{j_1},\ldots,x_{j_l}}$ through $(x_{i_1}, \ldots, x_{i_k})$ parallel to the universes $X_{j_1}, \ldots, X_{j_l}$. Fig. 1 illustrates the projection procedure of a two-dimensional set $\mu_\Gamma(x_1,x_2)$ on the Y-axis.

The cylindrical extension is more or less the opposite of the projection and extends the original domain of a fuzzy set to a greater universe. With H representing a relation on Y, $Y = \times_{m=1}^{k} X_{i_m}$ and $X = \times_{i=1}^{\Omega} X_i$ (see Fig. 2), the cylindrical extension of H into X is defined as:

$$\text{cext}_X(H) = \int_X \mu_H(x_{i_1}, \ldots, x_{i_k})/(x_1, \ldots, x_\Omega)$$

When using a Zadeh t-norm, application of the cylindrical extension and projection to deduct the resulting statement Δ of the inference of a rule Γ and a fact Λ gives rise to the following membership function for $\vartheta_2^{'}$:

$$\mu_{\vartheta_2^{'}}(y) = \sup_x \left\{ \min\left(\mu_{\vartheta_1^{'}}(x,y), \mu_\Gamma(x,y) \right) \right\}$$

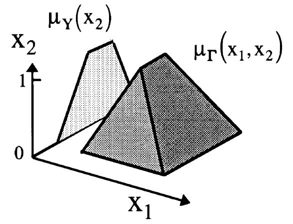


Fig. 1. Projection of $\mu_\Gamma(x_1,x_2)$ on Y.

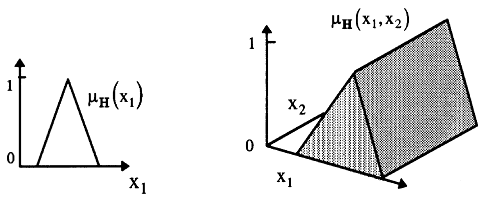


Fig. 2. Cylindrical extension of $\mu_H(x_1)$ into (x_1,x_2).

which embodies in essence the measure of compatibility of Λ with Γ.

3.2. The NEFCLASS algorithm

During a learning phase, NEFCLASS creates a fuzzy rule base by expressing every single rule in terms of a fuzzy implication. This rule base is then used in a test phase to classify new unseen cases by applying projection and cylindrical extension between every test case Δ and the existing set of classification rules Π. This latter operation leads to an activation measure for every rule Γ in Π which indicates how strongly a presented test case matches the premise part of Γ. NEFCLASS will classify the case at issue into the class which corresponds to the conclusion part of the rule with the uppermost activation. Prior to the very beginning of the learning algorithm an empty neural network is set up with as many input neurons as there are features in the dataset, as many output neurons as there are different class labels and zero hidden neurons. Aside from the construction of the neural framework a specific collection of fuzzy sets is defined for each feature. Fuzzy sets for a specific attribute are all of a triangular type and correspond to feature-related linguistic concepts such as ‘‘small’’, ‘‘medium’’ and ‘‘large’’ (Fig. 3).

Fig. 4 shows a fuzzy three-layer Perceptron NEFCLASS network after the execution of the learning algorithm. Since the network contains two input units, four hidden neurons and two output cells, it must have been trained on a dataset with n-tuples $x^{(n)}$, $n = 2$ that belong to a certain class ω_i out of the set $\{\omega_1,\omega_2\}$. For every input neuron α_i is determined a group of membership functions $\mu_{\chi_j}^{\alpha_i}(t),\ j \in \{1 \ldots \Phi_{\alpha_i}\}$ which nuance different linguistic concepts on the corresponding feature domain. By associating a single rule for every neuron (λ) on the hidden layer, adjusting its premise part to a conjunction of fuzzy sets on input to λ unit junctions and setting its conclusion part to the sole present λ to output unit link, one obtains the full set of classification rules that enclose the entire NEFCLASS network. If one equates ϕ_{α_i} to 3 $\forall i$, it can be seen in Fig. 4 every predefined membership function is (accidentally) preserved in at least one rule r_i, $i \in \{1,2,3,4\}$.

The establishment of a network structure takes place during the learning algorithm of NEFCLASS. This algorithm is split up into two subroutines: the construction of an initial rule base and the tuning of fuzzy set membership functions.

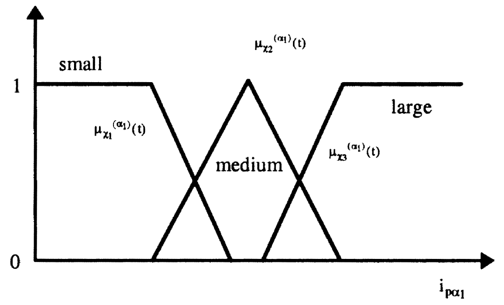


Fig. 3. Fuzzy sets and associated linguistic concepts for a particular input neuron.

3.2.1. Construction of an initial rule base

During the rule base construction phase, patterns are presented to the algorithm, hidden neurons are created and attached to a particular output unit and fuzzy sets are placed on connections between input units and hidden nodes. The rule base learning algorithm mainly consists of five different steps:

1. propagate the next pattern (i_p,t_p) through the network where t_p covers the desired network output for pattern p. $\forall$ input units $\alpha_i \in I$, find $\mu_{\chi_{\eta(\alpha_i)}}^{(\alpha_i)}(t)$ with $\mu_{\chi_{\eta(\alpha_i)}}^{(\alpha_i)}(t) = \max_{\delta \in \{1,\ldots,\phi_{\alpha_i}\}} \{\mu_{\chi_\delta}^{(\alpha_i)}(t)(a_{p\alpha_i}^{(0)})\}$
2. when no rule unit ν_h exists with $W_{\alpha_1\nu_h}^{(1)}(t) = \mu_{\chi_{\eta(\alpha_1)}}^{(\alpha_1)}(t)$, $W_{\alpha_2\nu_h}^{(1)}(t) = \mu_{\chi_{\eta(\alpha_2)}}^{(\alpha_2)}(t), \ldots,$ $W_{\alpha_{\tilde{i}}\nu_h}^{(1)}(t) = \mu_{\chi_{\eta(\alpha_{\tilde{i}})}}^{(\alpha_{\tilde{i}})}(t)$ create a new rule unit and connect it with output neuron β_o if $t_{p\beta_o} = 1$.
3. return to 1 as long as there remain unprocessed input/output pairs
4. the final rule base is determined by means of the following routine: steps 1 through 4 are executed without an upper bound on the maximum number of rules in the network.

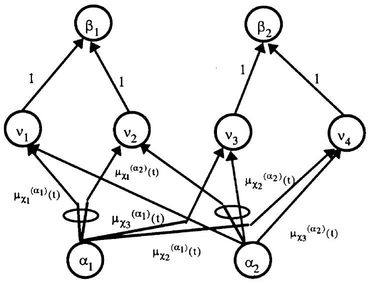
When all patterns have been sent through the network, one calculates for each separate class the accumulated activation (aa) of every single rule unit by propagating every pattern once more through the network. If the aa value for a rule unit for a certain class label ω_j exceeds the value for the label ω_i out of the conclusion part of the rule, the conclusion is altered from ω_i to ω_j. After modifying some conclusions, all patterns are presented to the network a third time and one calculates now for each rule unit the value:

$$v_{\nu_h} = \sum_p a_{p\nu_h}^{(1)}.e_p$$

where $a_{p\nu_h}^{(1)}$ = activation of rule unit ν_h for pattern p and

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

The final rule base is then constructed by keeping $k_{\max}$ hidden units with the highest v_{ν_h} value. In that way, one effectively avoids the learning process becoming dependent on the sequence of presented patterns. Fig. 5 illustrates the possible structure of a NEFCLASS network after processing two patterns.


β_1
β_2
1
1
1
1
ν_1
ν_2
ν_3
ν_4
$\mu_{\chi 1}^{(\alpha_1)}(t)$
$\mu_{\chi 1}^{(\alpha_2)}(t)$
$\mu_{\chi 2}^{(\alpha_2)}(t)$
$\mu_{\chi 3}^{(\alpha_1)}(t)$
$\mu_{\chi 3}^{(\alpha_2)}(t)$
$\mu_{\chi 2}^{(\alpha_1)}(t)$
α_1
α_2

r_1: if $i_{p\alpha_1}$ equals $\mu_{\chi 1}^{(\alpha_1)}(t)$ $\wedge$ $i_{p\alpha_2}$ equals $\mu_{\chi 1}^{(\alpha_2)}(t)$ then ω_1

r_2: if $i_{p\alpha_1}$ equals $\mu_{\chi 1}^{(\alpha_1)}(t)$ $\wedge$ $i_{p\alpha_2}$ equals $\mu_{\chi 2}^{(\alpha_2)}(t)$ then ω_1

r_3: if $i_{p\alpha_1}$ equals $\mu_{\chi 3}^{(\alpha_1)}(t)$ $\wedge$ $i_{p\alpha_2}$ equals $\mu_{\chi 2}^{(\alpha_2)}(t)$ then ω_2

r_4: if $i_{p\alpha_1}$ equals $\mu_{\chi 2}^{(\alpha_1)}(t)$ $\wedge$ $i_{p\alpha_2}$ equals $\mu_{\chi 3}^{(\alpha_2)}(t)$ then ω_2

Fig. 4. NEFCLASS network (left) and its corresponding rule base (right).

3.2.2. The fuzzy set learning algorithm

After NEFCLASS has produced a rule base, fuzzy sets on input to hidden unit connections are tuned to optimize classification results. The fuzzy set learning algorithm of NEFCLASS is a variant of the well-known back-propagation algorithm to train multi-layer Perceptron networks by altering weights on neuron to neuron junctions. Since non-differentiable t-norms are used as activation functions for hidden units, direct application of classical back-propagation to fuzzy NEFCLASS systems is impossible. Therefore, one has to forge a steepest descent heuristic which adjusts fuzzy membership functions on the basis of the fuzzy error of an output unit β_o:

$$E_{p\beta_o} = 1 - e^{\left[-\varepsilon(t_{p\beta_o} - o^{(2)}_{p\beta_o})^2\right]}$$

where $t_{p\beta_o}$ is the desired output of output unit β_o and ε is the error sensitivity.

The fuzzy error back-propagation heuristic proceeds then as follows:

1. Send the next input/output pair (i_p, t_p) through the network.
2. Calculate for each output neuron the value $\delta_{p\beta_o} = \mathrm{sgn}(t_{p\beta_o} - o^{(2)}_{p\beta_o})E_{p\beta_o}$ where sgn stands for a signum transfer function that takes on the value 1 or -1 depending on the sign of the argument.
3. For each rule unit whose output > 0:

- calculate:

$$\delta_{p\nu_h} = o^{(1)}_{p\nu_h}(1 - o^{(1)}_{p\nu_h}) \sum_o W^{(2)}_{\nu_h\beta_o}(t)\delta_{p\beta_o}$$

- find $\alpha_i \in I$ so that $W^{(1)}_{\alpha_i\nu_h}(t)(o^{(0)}_{p\alpha_i}) = \min_{i'}\{W^{(1)}_{a_{i'}\nu_h}(t)(o^{(0)}_{p\alpha_{i'}})\}$
- modify the structure of the fuzzy set $W(\alpha_i, \nu_h)(t)$ by means of the following delta-values (σ_a, σ_b and σ_c represent different learning rates):

$$\begin{aligned}
\delta_{pb} &= \sigma_a.\delta_{p\nu_h}.(c-a).\mathrm{sgn}(o^{(0)}_{p\alpha_i} - b) \\
\delta_{pa} &= -\sigma_b.\delta_{p\nu_h}.(c-a) + \delta_{pb} \\
\delta_{pc} &= \sigma_c.\delta_{p\nu_h}.(c-a) + \delta_{pb}
\end{aligned}$$

The above delta-values indicate the mutations that will have to be applied to the structure points a, b and c.

4. When a full cycle has been processed (all patterns are propagated through the network) and some stopping condition is met, then stop; otherwise continue with step 1 (see Fig. 6 for an example).

4. Results

In this section, we will compare the performance of C4.5 and NEFCLASS on a group of seven datasets which are taken from the UCI repository at Irvine, CA. Two experiments were conducted. In the first experiment, both algorithms were compared on the original datasets. The only pre-processing done was the removal of the unknown values from the datasets. In the second experiment, some additional pre-processing (feature selection and discretization) was performed before running the classification algorithms. The accuracy percentages were achieved by performing a stratified 10-fold cross validation.

4.1. Before discretization and feature selection

In Table 1 results of C4.5 and NEFCLASS on the training set and on the test set are depicted. Also the number of rules used to classify the objects can be found in the table.

A quick comparison reveals that on average the performance of C4.5 is better. Furthermore, the incapability of NEFCLASS to build compact rule bases can be concluded. If one takes a closer look at the rule base construction phase of NEFCLASS, the inevitability of encountering classification rule abundance becomes clear. If one defines three

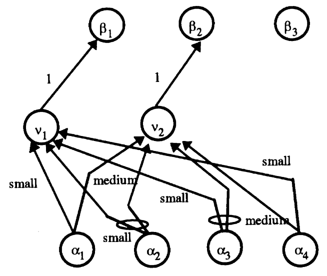


Fig. 5. Possible network structure after processing two patterns.

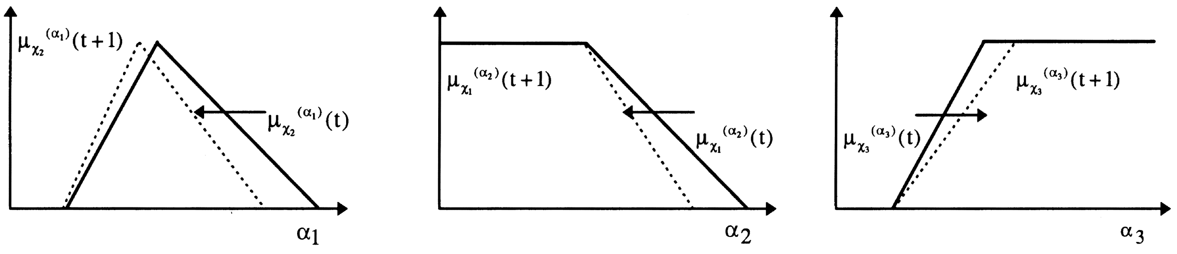

$\mu_{\chi_2}^{(\alpha_1)}(t+1)$
$\mu_{\chi_2}^{(\alpha_1)}(t)$
α_1
$\mu_{\chi_1}^{(\alpha_2)}(t+1)$
$\mu_{\chi_1}^{(\alpha_2)}(t)$
α_2
$\mu_{\chi_3}^{(\alpha_3)}(t+1)$
$\mu_{\chi_3}^{(\alpha_3)}(t)$
α_3

Fig. 6. Possible tuning scenarios of some membership functions.

linguistic concepts for every feature out of a dataset containing y 10-tuples $\mathbf{x}^{(10)}$, the total number of rules might come very close to y since generally 3^{10} is much higher than y. In fact, it is easy to see rule base magnitude will be a decreasing function of global dataset entropy, and hence of the overall degree of attribute to attribute correlations. In that respect, great care should be taken when adjudging NEFCLASS of composing complex collections of rules since the results below are retrieved by imposing no restrictions at all on the number of classification rules. Moreover, several experiments have been conducted in which a decreasing upper bound was instated on the extent of generated NEFCLASS rule bases.

Fig. 7 shows (full lines represent results on training data, dashed lines give the performance on test data) rule bases can be shrunk for some datasets (like breast and pima) to the level of 5% of their original size without a substantial loss in accuracy while for other datasets (like heart) rule base contractions lead to a collapse in performance.

In the context of the preceding phenomena, it seems imperative we integrate the NEFCLASS algorithm with some kind of auxiliary processing to create acceptable rule base magnitudes without causing global downfalls in accuracy for particular datasets.

4.2. After discretization and feature selection

In the second experiment we conducted, it was investigated how the results were influenced by means of discretization (the process which transforms continuous features in nominal ones by splitting up the domain of the feature in non-overlapping partitions) and feature selection (selecting the relevant subset of features). Both discretization and feature selection reduce the possible hypothesis space and so make it easier for the learning algorithm to classify new instances. It may seem strange that these simplifications may result in better classifiers because general opinion has it that it is best to collect as many data as possible, and subsequently let the learning system pick the relevant attributes. Of course, if an ideal algorithm should be used, this algorithm's performance should not degrade if more data are present. In practice, however, it has been shown that the performance of most learning algorithms degrades severely when too many irrelevant features are present in the data. Many discretization and feature selection and discretization algorithms exist in the literature. In this experiment, we used the discretization algorithm of Fayyad and Irani (1993) and the feature selection algorithm proposed by Kohavi (1995). As can be seen in Table 2, minor accuracy differences between C4.5 and NEFCLASS exist for some datasets (breast, heart, monks2 and pima) while huge contrasts can be observed for others (aucrx, monks1, monks3).

On average, C4.5 still outperforms NEFCLASS although the difference is smaller compared to the results without discretization and feature selection. Another observation we can make is that although the accuracy attained after discretization and feature selection is even higher, the number of rules needed to make a classification is reduced. This reduction is rather small for C4.5, but for NEFCLASS a very large reduction is obtained. This reduction is very important if we want to explain to the user the decision taken in a given situation because compactness of a classifier can greatly enhance its comprehensibility. For example, in Wets et al. (1997) it was demonstrated how after discretization and feature selection decision tables can be used to effectively visualize the extracted knowledge to the user.

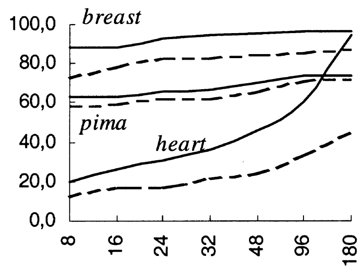


Fig. 7. Accuracy as a function of rule base extent.

Table 1
Results before discretization and feature selection

Data	Acc. training data		Acc. test data		No. of rules	
	C4.5	NEF.	C4.5	NEF.	C4.5	NEF.
Aucrx	91	89.4	98.7	87.5	12	283
Breast	97.1	95.1	94.7	95.6	16	218
Heart	91	92.0	77	61	13	237
Monks1	100	86.6	100	81.6	22	208
Monks2	88.4	86.8	69.8	83.4	34	211
Monks3	99	84.3	98	80.8	12	210
Pima	68.6	74.4	63.3	73.4	9	163
AVG	90.7	86.9	85.9	80.5	16.9	218.6

Table 2
Results after discretization and feature selection

Data	Acc. training data		Acc. test data		No. of rules	
	C4.5	NEF.	C4.5	NEF.	C4.5	NEF.
Aucrx	87	85.4	85	95.4	3	22
Breast	96.7	97	96.5	95.2	5	18
Heart	83.8	83.8	82.7	82.5	8	23
Monks1	100	83.6	100	83.0	22	18
Monks2	99.9	100	96.6	98	40	61
Monks3	96.5	81.5	95.7	81.5	8	6
Pima	78.4	74.4	77.9	71.4	6	13
AVG	91.8	86.5	90.6	86.7	13.1	23.0

5. Conclusions

In this paper, two existing classifiers (C4.5 and NEFCLASS) have been studied. While the C4.5 program is a typical machine learning induction algorithm, the NEFCLASS strategy combines fundamental aspects of the domain of fuzzy set theory with a multi-layer neural Perceptron network. Based on the outcome of our experiments, it can be concluded that on average C4.5 outperforms NEFCLASS before and after discretization and feature selection. However, it is important to note that on some datasets NEFCLASS performs better than C4.5, while on others it does a lot worse. A topic for further research will be to investigate whether specific characteristics of data can be found, which explain this behaviour.

Secondly, we can conclude that discretization and feature selection improved the attained accuracy of both classifiers and the modifications. The experiments have unveiled the substantial reduction in rule base magnitude that can be obtained by pre-processing both C4.5 and NEFCLASS with discretization and feature selection. This is very important if we not only want to use the rules for classification purposes but also to explain to the user why a certain decision has been made.

References

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. New York: Plenum Press.

Fayyad, U., & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *Proceedings of the 13th international joint conference on artificial intelligence* (pp. 1023–1027).

Kohavi, R. (1995). The power of decision tables. *Proceedings of the European conference on machine learning* (pp. 174–189), Lecture Notes in Artificial Intelligence 914. Springer Verlag.

Kuncheva L.I. (1996). On the equivalence between fuzzy and statistical classifiers.. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 4, 246–249.

Nauck, D., Klawonn, F., & Kruse, R. (1996). *Neuronale netze und fuzzy systeme*. Vieweg Braunschweig/Wiesbaden.

Quinlan, J. R. (1993). *C4.5 programs for machine learning*, San Mateo, CA: Morgan Kaufmann.

Weber S. (1983). A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy Sets and Systems*, 11, 115–134.

Wets, G., Piramuthu, S., & Vanthienen, J. (1997). Extending a tabular knowledge based framework with feature selection. *Expert Systems with Applications*, *13*, to appear.

Zadeh L.A. (1965). Fuzzy sets. *Information and Control*, 8, 338–353.

Zadeh, L. A. (1979). A theory of approximate reasoning. *Machine intelligence 9* (pp. 149–194). New York: Halstead Press.

Zimmermann (1991). *Fuzzy set theory and its applications* (2nd ed.). Kluwer Academic.