Soft Comput (2007) 11:439–457
DOI 10.1007/s00500-006-0108-0

FOCUS

Intelligent data analysis with fuzzy decision trees

Xiaomeng Wang · Detlef D. Nauck · Martin Spott · Rudolf Kruse

Published online: 12 September 2006

Abstract Intelligent data analysis has gained increasing attention in business and industry environments. Many applications are looking not only for solutions that can automate and de-skill the data analysis process, but also methods that can deal with vague information and deliver comprehensible models. Under this consideration, we present an automatic data analysis platform, in particular, we investigate fuzzy decision trees as a method of intelligent data analysis for classification problems. We present the whole process from fuzzy tree learning, missing value handling to fuzzy rules generation and pruning. To select the test attributes of fuzzy trees we use a generalized Shannon entropy. We discuss the problems connected with this generalization arising from fuzzy logic and propose some amendments. We give a theoretical comparison on the fuzzy rules learned by fuzzy decision trees with some other methods, and compare our classifiers to other well-known classification methods based on experimental results. Moreover, we show a real-world application for the quality control of car surfaces using our approach.

X. Wang · R. Kruse (✉)
Faculty of Computer Science, University of Madgeburg,
Universitaetsplatz 2, 39106 Magdeburg, Germany
e-mail: kruse@iws.uni-magdeburg.de

X. Wang
e-mail: xwang@iws.uni-magdeburg.de

D. D. Nauck · M. Spot
BT, Research and Venturing,
Intelligent Systems Research Centre,
Adastral Park, Orion Bldg. pp1/12,
Ipswich IP5 3RE UK
e-mail: detlef.nauck@bt.com

M. Spot
e-mail: martin.spott@bt.com

Keywords Fuzzy decision trees · Intelligent data analysis · Classification models · Fuzzy rule learning

1 Introduction

Modern computer technologies take us into a new information age. Nowadays data from different sources and application fields can be gathered fast at low cost and often in enormous amounts. To understand hidden phenomena, answer questions and make decisions data alone are not sufficient, rather the intelligent analysis of the data is desired.

Intelligent data analysis (IDA) is a research area that has its roots in machine learning, statistics, soft computing and databases and looks for technologies and strategies to support and to (partially) automate the data analysis and knowledge detection process. According to the goals data analysis can be categorized as exploratory analysis (e.g. data visualization), descriptive analysis (e.g. segmentation, clustering techniques), predictive analysis (e.g. classification and regression), etc.

Our work is concerned with classification problems. Classical decision trees as one of the popular classification models, work well in crisp domains, but cannot model vagueness. Aiming at learning a classification model, which is comprehensible and able to handle vagueness, in this paper we study fuzzy decision trees by combining fuzzy theory with classical trees.

In order to push advanced data analysis technology into business, software that empowers users and hides complexity from them is needed. Section 2 is dedicated to our automatic data analysis platform SPIDA. We give the motivation of our work on automating IDA, state

the premises of our approach and introduce the architecture, features and functions of SPIDA. In Sect. 3 we present our fuzzy decision tree method, which was investigated and implemented for SPIDA. We examine in detail the attribute selection with extended information measures as well as the treatment of missing values. Section 3.7 shows the fuzzy rule base generated from the induced fuzzy tree, compares the rule learning process with some other methods, studies some heuristics to simplify the rule base and illustrates the classification of new data with the fuzzy rules. In Sect. 4 we compare our approach to some popular classifiers based on the experimental results. Section 5 presents a real world application.

2 A soft computing platform for automatic intelligent data analysis – SPIDA

With ever increasing amounts of collected data about internal processes and customer interactions, today's businesses are faced with the problem of extracting knowledge from that data. It becomes unsustainable and expensive to employ experts to conduct data analysis manually, therefore solutions that can automate and de-skill the data analysis process are desirable.

Especially in industrial settings it is important to empower non-expert users in coping with daily data analysis tasks. From our experience take-up of traditional data analysis software is very slow. We believe that typical business users cannot use technology or method oriented software. Without detailed knowledge of the analysis methods such tools are basically useless. In order to push advanced data analysis technology algorithms into businesses we need software that empowers users and hides complexity from them.

From our point of view users need a tool where they can specify a data source and requirements on the solution of an analysis process. The tool can select appropriate methods and apply them automatically. The results are checked against the user requirements. If the requirements are not sufficiently met, the analysis is repeated with different parameters. At the end, the user is presented with a set of possible solutions and their descriptions in terms of user requirements. After selecting a solution this will be wrapped in a software module that can be readily applied in the user's application domain.

Our approach to automating IDA is based on the premises that user requirements, properties of analysis methods, and expert knowledge for the selection of the right method and its configuration are quite often fuzzy. For instance, users demand *fast, accurate, simple, inexpensive* solutions. Some analysis methods are *fast*, some can produce *accurate* results, others are *interpretable*. Finally, data analysis experts have in addition to their formal knowledge about analysis methods vague intuitive knowledge on how to select parameters and methods and how to run an analysis in a certain scenario. In order to deal with such vagueness we decided to use fuzzy representations for user requirements, properties of analysis methods and expert knowledge.

Previous approaches towards automating data analysis were mostly based on formal AI techniques that simply do not meet the above requirements for non-expert users. A discussion can be found in Nauck et al. (2004).

Architecture, features and functions of SPIDA

Based on the requirements set above the Computational Intelligence Group from BT Research & Venturing developed SPIDA (Soft computing platform for intelligent data analysis) (Nauck et al. 2003). Essentially, SPIDA is a data analysis tool comprising a set of data analysis methods mainly from the area of soft computing and related areas (neural networks, neuro-fuzzy systems, support vector machines, decision trees etc.), data filters for pre- and post-processing, visualization capabilities and access to different data sources (text files, databases).

The main user group targeted by SPIDA are domain experts. They are typically familiar with their data, they know the processes that produce the data, and are usually keen to review these processes in order to improve or understand them. Furthermore, gained knowledge can also be applied to other problems that are related to the data like using information gained from customer data for marketing purposes. Domain experts are usually no data mining experts, but they can specify their data analysis problem and their requirements for the solution at a high level. Based on this information and the data, the SPIDA Wizard selects and runs data analysis methods automatically.

Each analysis method that is available in SPIDA is described by fuzzy properties like interpretability, adaptability to new data, accuracy and non-fuzzy properties like analysis problem (classification, regression, clustering etc.). Similarly, users define their requirements using similar terms: the type of problem, if they need an explanation (interpretation) facility, the expected simplicity and type (rules, functions) of the explanation, adaptability, accuracy etc. These requirements will be mapped onto desired properties of the analysis method (model properties) using the fuzzy knowledge base for method selection (KBMS), see Fig. 1. Given an appropriate

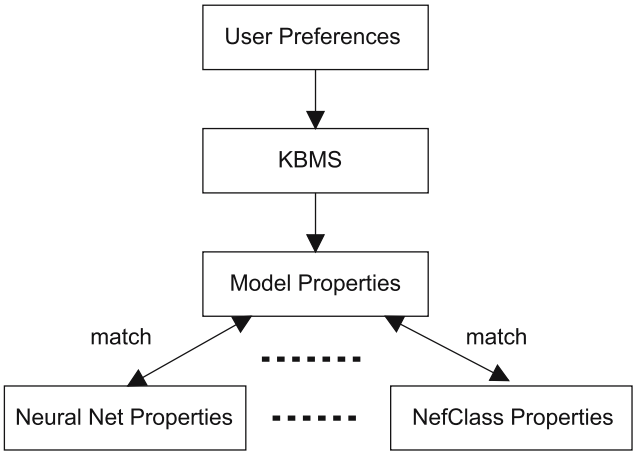

Fig. 1 Automatic selection of methods. Using the KBMS, wanted model properties are inferred from user preferences. Properties of available models are matched against wanted properties, which results in a ranking of models

KBMS, user requirements could be defined at an even higher level like 'I wish a model that predicts tomorrow's price of BT shares with high accuracy, it should be easy to understand using rules and I would like to implement it as an SQL query'.

The current version of the SPIDA Wizard provides a sequence of dialogs like the one shown in Fig. 2 to specify user requirements and the data source. It then selects all applicable analysis methods and ranks them according to the requirements. The user then selects the methods he is interested in and SPIDA creates so-called analysis workspaces for the chosen methods. Such a workspace consists of blocks for data access, filters, analysis methods and visualization as shown in Fig. 3. The configuration of the workspace and its blocks depends on the data, the analysis problem and the chosen analysis method. The workspaces will be executed, model properties like accuracy and simplicity will be measured and the suitability of the model reevaluated. In case of a low suitability, the Wizard tries to change the parameters of the analysis method and reruns the workspace in order to improve the results. An example of this procedure is shown in Fig. 4.

In summary, SPIDA consists of the following function blocks

- Wizard: user interface for non-expert users to run SPIDA in automatic mode.
- Automatic pre-processing: detection of data types, handling of missing values, normalisation, scaling, re-coding etc.

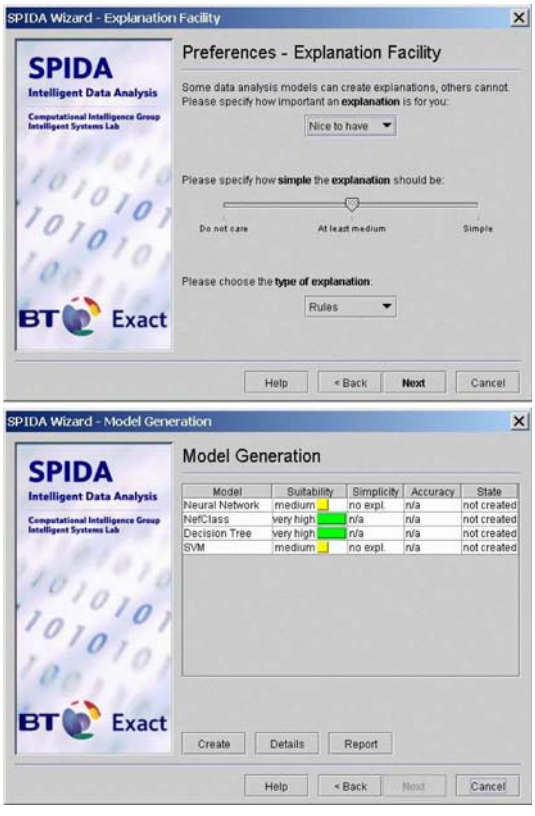

Fig. 2 Choosing preferences in the Wizard, *top* the dialog for an explanation facility of methods, *bottom* ranking of data analysis methods according to user preferences before model creation

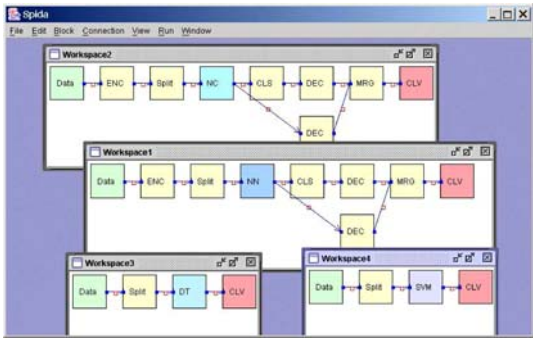

Fig. 3 Workspaces automatically created by the wizard for four different data analysis methods. The workspaces are usually not shown by the wizard. This view also represents the GUI an IDA expert would use to configure data analysis processes manually

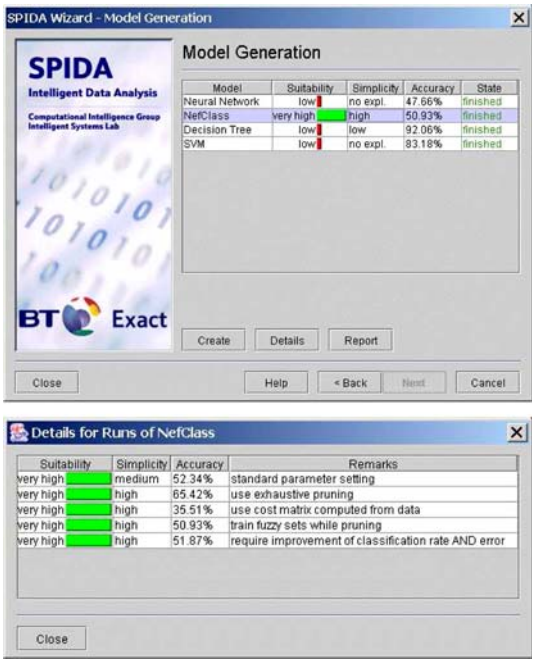

Fig. 4 *Top* results of the model creation process. *Bottom* automatic re-runs of the NEFCLASS model

- Automatic method selection: depending on the user requirements and the data to be analysed methods are selected. If a method requires changes to the data format pre-processing is invoked.
- Automatic method execution: IDA processes are configured and started. Method execution is monitored by the SPIDA knowledge base and methods are automatically re-configured or re-run if necessary.
- Adaptive user profiling: for each user a profile is maintained comprising user-specific requirements for IDA processes.
- Automatic result evaluation: this module compares results of IDA processes to (fuzzy) user requirements. If no sufficient match is obtained, the processes are re-configured and re-run to obtain a better match.
- Automatic solution generation: this module wraps the selected IDA results into executable objects with standardized interface, which can be used in user applications directly.
- Expert interface: experts can use SPIDA in a non-automatic expert mode, where IDA processes are graphically constructed and can be fully controlled.

For a more detailed description of the SPIDA Wizard that drives the automatic data analysis process see Nauck et al. (2003, 2004).

3 Fuzzy decision trees

The fuzzy decision tree (FDT) module here has been developed for SPIDA. In this implementation, the configuration for the underlying program is done through the GUI, which is running on the client; and the algorithm for model learning is resident on the server. If we consider the previous section as the presentation of automatic IDA on the higher level, where methods for a given problem can be selected automatically, then making each individual module easy to use can be regarded as automating IDA on the lower level. Our fuzzy decision tree module does not require many user interactions due to its user-friendly GUI, automatic fuzzy partitioning techniques and automatic pruning strategies (details see below).

In the past several variants of FDTs were introduced by different authors. Some work on binary fuzzy trees has been done in Boyen and Wehenkel (1995), Olaru and Wehenkel (2003) and Drobics and Bodenhofer 2002, where in Boyen and Wehenkel (1995)has presented the automatic induction of binary fuzzy trees based on a new class of discrimination quality measures. Janikow (1998) adapted the well-known ID3 algorithm so that it works with fuzzy sets. In this paper, we also adapted the ID3 algorithm to construct FDTs and borrowed some basic ideas from Janikow (1998).

One important issue in tree learning is to select test attributes. In existing FDT induction algorithms different measures for ranking attributes were proposed, for example, a squared error function in Olaru and Wehenkel (2003), a measure of classification ambiguity in Yuan and Shaw (1995), or simply the class entropy related to an attribute Bouchon-Meunier et al. (1996).[1] Still there exists a lot of work on fuzzy decision trees where a generalized Shannon entropy based information gain is applied, e.g. in Janikow (1998), Drobics and Bodenhofer (2002) and Mitra et al. (2002). In Sect. 3.4 we will discuss the problems occurring with the extended information gain and its successor information gain ratio in the fuzzy domain. Although we investigated the problems here only considering information gain as example, because it is the most widely used measure in decision tree learning, the observed problems are much more general: if some measures from classical trees are transferred into fuzzy decision trees in an incautious way, then not only positive but also negative values for these measures could be produced in the later case (we discuss this in Sect. 3.4). This will cause undesired effects in ranking

[1] A survey of different measures is given in Bouchon-Meunier and Marsala (1999). A comparison study of some measures can be found in Marsala and Bouchon-Meunier (2003).

attributes once a normalization on the measures is done (see the discussion on information gain ratio in Sect. 3.5).

Going beyond Janikow's work, in Sect. 3.7 we consider how to extract a fuzzy rule base from the resulted fuzzy tree and use this rule base to perform the classification.

3.1 FDT learning

Like classical decision trees with the ID3 algorithm, fuzzy decision trees are constructed in a top-down manner by recursive partitioning of the training set into subsets. In this paper we focus on the induction of a fuzzy decision tree on continuous attributes. Before the tree induction a fuzzy partition has to be created for each attribute. The fuzzy sets of these partitions will be used as fuzzy tests in the nodes of the fuzzy tree. To initialize these fuzzy partitions we adopted an existing algorithm, which creates them either completely automatically based on a given data set (called "automatic partitioning"), or based on a user specification of the shape and number of the membership functions (called "individual partitioning"). In the latter case the fuzzy sets are distributed evenly over the entire domain of each attribute. Here we assume the fuzzy partitions of the input variables are given.

To highlight the differences to the classical decision trees, we point out some particular features of fuzzy tree learning as follows:

1. The membership degree of examples
   The membership degree of an example to an example set is not a binary element from $\{0,1\}$ (as in classical decision trees), but from the interval $[0,1]$. In each node, an example has a different membership degree to the current example set, and this degree is calculated from the conjunctive combination of the membership degrees of the example to the fuzzy sets along the path to the node, where different t-norms ($\top$), e.g. $\top_{\text{Prod}}(a,b) = a \cdot b$, $\top_{\min}(a,b) = \min(a,b)$ or $\top_{\text{Łukasiewicz}}(a,b) = \min\{1, 1-a+b\}$ with $0 \le a,b \le 1$, can be used for this combination.
2. Selection of test attributes
   This point will be discussed in detail in the following subsections.
3. Fuzzy tests
   As mentioned above, in the inner nodes fuzzy tests are used instead of crisp tests. A fuzzy test of an attribute means to determine the membership degree of the value of an attribute to a fuzzy set.
4. Leaf labeling
   There are two variants to label a leaf node: each leaf node contains all classes including the corresponding membership degrees, or each leaf is labeled with the class having the largest membership degree. Here we use the first variant (see Sect. 3.3).
5. Stop criteria
   Usually classical tree learning is terminated if all attributes are already used on the current path; or if all examples in the current node belong to the same class. Here we use an additional condition, namely whether the information measure is below a specified threshold. In FDT any example can occur in any node with any membership degree. Thus in general more examples are considered per node and fuzzy trees are usually larger than classical trees. The threshold defined here enables a user to control the tree growth, so that unnecessary nodes are not added. The experiments prove that an adequate threshold helps not only to avoid overfitting, but also to decrease the complexity of the tree.

3.2 Notation

For the formal presentation of a FDT learning and discussion of attribute selection we introduce the following notation:

1. $A = \{A_1, A_2, \ldots, A_n\}$ is a set of input attributes with domains $\text{dom}(A_i)$, $1 \le i \le n$.
2. For each variable $A_i \in A$, $1 \le i \le n$:
   - $u^i \in \text{dom}(A_i)$ is a crisp value of attribute A_i.
   - D_i is the fuzzy partition of A_i.
   - a_p^i denotes the fuzzy set (the linguistic term) p for the attribute A_i. For example $a_{\text{low}}^{\text{pressure}}$ means the fuzzy set "low" of the attribute "pressure".
3. $C = \{C_1, \ldots, C_m\}$ is the set of possible classes.
4. We consider a reference set $E = \{e_1, \ldots, e_s\}$ of examples $e_k = (\mathbf{u}_k, \mathbf{y}_k)$, $1 \le k \le s$. $\mathbf{u}_k$ is the input vector, $\mathbf{y}_k \in [0,1]^m$ the output vector of e_k. u_k^i (i.e. the ith element of $\mathbf{u}_k$) is the (crisp) value of attribute A_i in example e_k. y_k^j (i.e. the jth element of $\mathbf{y}_k$) is the membership degree of example e_k to class C_j.
   The crisp classification of e_k can be computed as $\text{class}(e_k) = \text{argmax}_{j:1\le j\le m}\{y_k^j\}$. (Note that a classification problem can easily be extended for a continuous target variable by using fuzzy sets instead of crisp classes C_j. However, we confine ourselves to targets with a finite number of classes.)
5. The training set is a fuzzy set over E defined by initial confidence weights $\chi = \{\chi_1, \ldots, \chi_s\}$, $\forall k, 1 \le k \le s : 0 \le \chi_k \le 1$. If no such information is provided by a user, we set $\forall k, 1 \le k \le s : \chi_k = 1$.

Fig. 5 FDT learning algorithm

Algorithm 1.
FDT(*examples*$\{e_k\}$, *training set*$\{\chi_k\}$, *class_variable*, *attributes*)

- Create a *Root* node for the tree
- If all examples belong to one class, return *Root*
- If *attributes* is empty, return *Root* with the membership degrees to the classes
- Otherwise begin
  - Calculate the information measure for each attribute (call **Algorithm 2.**, see Figure 6)
  - $A \leftarrow$ the "best" attribute according to the highest information measure, if this information measure is below the threshold specified by the user, return root; else continue
  - The test attribute for *Root* $\leftarrow A$
  - For each fuzzy set a_p of A
    - Update training set $\{\chi_k\}$ into $\{\chi_k^{\cdot|a_p}\}$ according to the membership degree to a_p as: $\chi_k^{\cdot|a_p} = \top(\chi_k, \mu_{a_p}(u_k^A))$, where $\mu_{a_p}(u_k^A)$ is the membership degree of attribute A's value u_k^A in example e_k to fuzzy set a_p
    - If all elements of $\{\chi_k\}$ are equal 0, go to the next fuzzy set a_{p+1} without adding a new branch; else add a new branch below *Root* connecting with the test $A = a_p$ and below this new branch add the subtree FDT(*examples*$\{e_k\}$, *training set*$\{\chi_k^{\cdot|a_p}\}$, *class_variable*, *(attributes-*$\{A\}$*)*)
- Return *Root*

6. For each node N in the fuzzy tree, $\chi^N = \{\chi_1^N, \ldots, \chi_s^N\}$ is the fuzzy example set (a fuzzy set over E) in N. In the root, this fuzzy example set coincides with the training set, i.e., $\forall k, 1 \le k \le s$: $\chi_k^{\text{Root}} = \chi_k$.
7. $Z_{C_j}^N = \sum_{k=1}^{s} \top(\chi_k^N, y_k^j)$ stands for the example counter for class C_j in node N, where $\top$ is a t-norm. $Z^N = \sum_{j=1}^{m} Z_{C_j}^N$ is the total counter for the examples of all classes. (More details about the computation and the interpretation of membership degrees as case weights are given in Sect. 3.4.).
8. $I(\chi^N)$ denotes the generalized Shannon entropy of the class distribution w.r.t. the fuzzy example set χ^N in node N. $I(\chi^N|A_i)$ is the weighted sum of entropies from all child nodes, if A_i is used as the test attribute in node N.
9. $\text{Gain}(\chi^N, A_i) = I(\chi^N) - I(\chi^N|A_i)$ is the information gain w.r.t. attribute A_i, which is the first of the two attribute selection measures we consider.
10. $\text{SplitI}(\chi^N, A_i)$ denotes the split information – the entropy w.r.t. the value distribution of attribute A_i (instead of the class distribution).
11. $\text{GainR}(\chi^N, A_i) = \text{Gain}(\chi^N, A_i)/\text{SplitI}(\chi^N, A_i)$ is the information gain ratio w.r.t. attribute A_i, which is the second attribute selection measure we consider.

3.3 Learning algorithm

The algorithm for learning FDTs is shown in Fig. 5.

Based on the notation introduced in the previous section, we now present how the class information in each leaf is determined for the labeling. There exist various ways to do this, here we use a simple one: $Z_{C_j}^N = \sum_{k=1}^{s} \top(\chi_k^N, y_k^j)$ denotes the example counter for class C_j in leaf node N, then each class (with its membership degree) can be described as a fuzzy singleton $(C_j : w_j)$, where

$$w_j = \frac{Z_{C_j}^N}{\text{argmax}_{l:1\le l\le m}\{Z_{C_l}^N\}} \quad \text{with } 1 \le j \le m.$$

3.4 The problems of information measures in the fuzzy domain

The central point during tree learning is to select the "best" test attribute according to some information measures. Two well-known measures in classical decision tree induction – information gain and information gain ratio – were introduced by Quinlan in (1986) and (1993), respectively. The attribute yielding the highest information gain or gain ratio is chosen for the test.

The definition of information gain is based on probability theory, more exactly on Shannon entropy. Information gain was used in ID3 for a long time and produced very good results. However, this measure has an inherent bias in favoring attributes with many values. A simple example explains this bias clearly: if a data set has an attribute, which records the identification number of each data example, the information gain of this attribute will be the highest, because with this attribute the data set will be split into so many subsets with each containing only a single data example. This leads to a tree that although classifies the learning data perfectly, obviously makes very poor prediction on new data. To solve this problem, a normalization of information gain in C4.5 Quinlan (1993) was proposed, namely information gain ratio: $\mathrm{GainR}(E,A) = \mathrm{Gain}(E,A)/\mathrm{SplitI}(E,A)$, where E is a learning data set, A an attribute, $\mathrm{Gain}(E,A)$ information gain of attribute A and $\mathrm{SplitI}(E,A)$ the entropy of E with respect to the values of attribute A. Since the attribute with many values has not only a high information gain but also a high entropy, the normalization brings the desired compensation effect, i.e., compensates the bias of information gain.

In the following, we discuss the problems that occur if we apply these two measures in fuzzy decision tree induction. We used the generalized Shannon entropy as in Janikow (1998): recall the notation in Sect. 3.2, and suppose $\chi^N = \{\chi_1^N, \ldots, \chi_s^N\}$ $(1 \le k \le s)$ is the fuzzy example set in node N in the fuzzy tree. Then the generalized entropy of χ^N is computed as

$$I(\chi^N) = -\sum_{j=1}^{m} \left(\frac{Z_{C_j}^N}{Z^N}\right) \log_2 \left(\frac{Z_{C_j}^N}{Z^N}\right), \tag{1}$$

where $Z_{C_j}^N$ is the example counter for class C_j $(1 \le j \le m)$ and Z^N is the total counter for the examples of all classes in node N.

Suppose attribute A (with $|D|$ fuzzy sets $\{a_p | 1 \le p \le |D|\}$) is the test attribute in N. Then the generalized average entropy of children corresponding to A is computed as:

$$I(\chi^N|A) = -\sum_{p=1}^{|D|} \frac{Z^{N|a_p}}{\hat{Z}^N} \sum_{j=1}^{m} \left(\frac{Z_{C_j}^{N|a_p}}{Z^{N|a_p}}\right) \log_2 \left(\frac{Z_{C_j}^{N|a_p}}{Z^{N|a_p}}\right), \tag{2}$$

where $Z_{C_j}^{N|a_p}$ is the counter for the examples that belong to fuzzy set a_p and class C_j, $Z^{N|a_p}$ the counter for all examples in child node $N|a_p$, $\hat{Z}^N$ the counter for the entire set of examples coming from all children, i.e.: $\hat{Z}^N = \sum_{p=1}^{|D|} Z^{N|a_p}$ (in the fuzzy domain $\hat{Z}^N$ can be different from Z^N).

The difference of $I(\chi^N)$ and $I(\chi^N|A)$ yields the information gain w.r.t. A. Clearly, the definition of information gain in the fuzzy domain differs from the classical one mainly in the computation of the various counters listed above, because in fuzzy decision trees, the counter does not refer to the number of the examples but rather the sum of the case weights.

As stated in feature 1 in Sect. 3.1, each example e_k in node N is associated with a membership degree χ_k^N, which is calculated on the basis of the conjunctive restrictions along the path from the root to node N and computed incrementally with t-norms. Once a test attribute is chosen and the child nodes are added, new membership degrees of the examples in each child must be computed, because new restrictions represented by the fuzzy sets of the test attribute are added. For example, the membership degree of example e_k to the fuzzy example subset corresponding to fuzzy set a_p (namely child node $N|a_p$) is calculated as $\chi_k^{N|a_p} = \top_1(\mu_{a_p}(u_k^A), \chi_k^N)$. Therefore, the counter for the examples of class C_j in node $N|a_p$ is $Z_{C_j}^{N|a_p} = \sum_{k=1}^{s} \top_2\left(\chi_k^{N|a_p}, y_k^j\right)$, and the counter for the examples of all classes is $Z^{N|a_p} = \sum_{j=1}^{m} Z_{C_j}^{N|a_p}$. Hence $\left(Z_{C_j}^{N|a_p}/Z^{N|a_p}\right)$ means the probability of class C_j given the multi-dimensional fuzzy set defined by the fuzzy sets along the path from the root to node $N|a_p$.

In classical cases, the value of information gain, accordingly the value of information gain ration, as we know, can never be negative (a prove can be found, for instance, in Borgelt and Kruse (2002)). The non-negative feature of information gain and gain ratio relies on the premise, that after splitting the sum of example cases belonging to each class in the union of all subsets is the same as the corresponding sum before splitting, namely the class frequency distribution in the whole set is retained after splitting. In other words, the definition of information gain implicitly assumes that this premise is satisfied. In fact, this premise is guaranteed in classical probabilistic theory. However, in FDT the computation is based on the weights of example cases, not simply the number of cases. Depending on the computation, negative information gain is possible in FDT. This phenomenon occurs due to the following two reasons, both of which can lead to a situation in which the sum of the weights of example cases before and after a split and thus the class frequency distributions differ.

1. In fuzzy logic, the sum of the membership degrees of a value to the fuzzy sets of its variable can differ from 1, depending on how the fuzzy sets overlap.

2. Probability theory prescribes to use the product to express a (conditional) conjunction (i.e., $P(X \wedge Y) = P(X \mid Y) \cdot P(Y)$), whereas fuzzy logic offers other possibilities besides the product, for example $\top_{\min}(a,b) = \min(a,b)$ and $\top_{\text{Łukasiewicz}}(a,b) = \min\{1, 1-a+b\}$ with $0 \le a,b \le 1$.

With the following examples we show how fuzzy logic can influence information measures.

Example illustrates negative information measures due to membership degrees

- Let $E = \{e_1, e_2, e_3, e_4, e_5\}$ be a reference set with examples coming from two classes C_1 and C_2, where the membership degree of the kth example to the jth class y_k^j have the following values: $y_1^1 = y_2^1 = 1$, $y_1^2 = y_2^2 = 0$ (i.e., e_1, e_2 belong exclusively to C_1), and $y_3^1 = y_4^1 = y_5^1 = 0$, $y_3^2 = y_4^2 = y_5^2 = 1$ (i.e., e_3, e_4, e_5 belong exclusively to C_2).
- Let the membership degree of each example to the current fuzzy example set in node N be $\chi_k^N = 1$, $1 \le k \le 5$ (see Table 1). These membership degrees are interpreted as case weights and thus we have $p_{C_1} : p_{C_2} = 2 : 3$ as the frequency distribution of the classes.
- After splitting the training set according to the fuzzy sets of attribute A — *small* and *large* — we obtain Table 2 showing the membership degree of each example. Since $\chi_k^N = 1$, $1 \le k \le 5$, if $\top_{\text{Prod}}$ is used, the membership degree of each example is the same as its membership degree to the respective fuzzy set,[2] e.g. $\chi_1^{N|\text{small}} = \top_{\text{Prod}}(\chi_1^N, 0.8) = 0.8$ and $\chi_1^{N|\text{large}} = \top_{\text{Prod}}(\chi_1^N, 0.6) = 0.6$. If we interpret the membership degrees to the fuzzy example set as case weights we can sum the weights for the subsets to obtain the weights for the whole set (as it is possible in classical decision tree induction). In this way we obtain case weights (**not** membership degrees, because they may be greater than 1) $\hat{\chi}^N$, for which we have $\hat{\chi}_1^N = 0.8 + 0.6 = 1.4$, $\hat{\chi}_2^N = 0.7 + 0.9 = 1.6$, and $\hat{\chi}_3^N = \hat{\chi}_4^N = \hat{\chi}_5^N = 1$. The frequency distribution of the classes w.r.t. these case weights is $p_{C_1} : p_{C_2} = (1.4 + 1.6) : (1.0 + 1.0 + 1.0) = 3 : 3 = 1 : 1$.

Table 1 Example 1 (membership degrees of examples before split)

Before split	C_1		C_2		
	e_1	e_2	e_3	e_4	e_5
χ_j^N	1.0	1.0	1.0	1.0	1.0
Z^N	2.0			3.0	

Table 2 Example 1 (membership degrees of examples after split)

After split	C_1		C_2		
	e_1	e_2	e_3	e_4	e_5
Small	0.8	0.7	1.0	0	1.0
Large	0.6	0.9	0	1.0	0
$\hat{\chi}_j^N$	1.4	1.6	1.0	1.0	1.0
$\hat{Z}^N$	3.0			3.0	

Obviously the sum of the case weights has changed after splitting the training set according to the fuzzy partition of attribute A. Since the entropy of a uniform probability distribution is maximal, the entropy $I(\hat{\chi}^N)$ after the split is certainly larger than the entropy $I(\chi^N)$ before the split. If we have $\forall k, 1 \le k \le 5 : \hat{\chi}_k^N = \chi_k^{N|\text{small}} + \chi_k^{N|\text{large}}$ (as it is implicitly assumed by the definition of information gain), we have that $\widehat{\text{Gain}}(\chi^N, A) = I(\hat{\chi}^N) - I(\chi^N|A)$ is definitely non-negative. Now it is easy to conclude:

$$\begin{aligned} &\text{since } I\left(\hat{\chi}^N\right) > I\left(\chi^N\right) \\ &\quad \text{and } \widehat{\text{Gain}}\left(\chi^N, A\right) = I\left(\hat{\chi}^N\right) - I\left(\chi^N|A\right) > 0 \\ &\quad \Longrightarrow \text{Gain}\left(\chi^N, A\right) = I\left(\chi^N\right) - I\left(\chi^N|A\right) \\ &\quad < 0 \quad \text{is possible.} \end{aligned}$$

Consequently, $\text{GainR}(\chi^N, A) = \text{Gain}(\chi^N, A)/\text{SplitI}(\chi^N, A) < 0$ is possible, since $\text{SplitI}(\chi^N, A)$ is non-negative. Indeed, for the example considered above we have $\text{Gain}(\chi^N, A) = 0.971 - 0.98 = -0.008$. An illustration for negative information gain caused by $\top_{\min}$ can be easily constructed in analogy to Example 1 as follows.

Example illustrates negative information measures caused by $\top_{\min}$

- Table 3 shows a reference set $E = \{e_1, e_2\}$ coming from two classes C_1 and C_2 in node N, where e_1 belongs exclusively to C_1 with $y_1^1 = 1$ (i.e. $y_1^2 = 0$) and e_2 exclusively to C_2 with $y_2^2 = 1$ (i.e. $y_2^1 = 0$).
- Assume the membership degrees of these two examples to the current fuzzy example set are: $\chi_1^N = 0.6$ and $\chi_2^N = 0.9$. These membership degrees are interpreted as case weights and thus the frequency distribution of the classes is $p_{C_1} : p_{C_2} = 2 : 3$.

[2] In general, one has to combine the membership degree to the fuzzy example set and the membership degree to the fuzzy set of the attribute with a t-norm.

Table 3 Example 2 (membership degrees of examples before split)

Before split	C_1	C_2
	e_1	e_2
χ_j^N	0.6	0.9
$\hat{Z}^N$	0.6	0.9

- Attribute A is the test attribute and has two fuzzy sets small and large. Assume the value of A in e_1 has membership degrees $\mu_{\text{small}}(e_1) = 0.6$ to fuzzy set small and $\mu_{\text{large}}(e_1) = 0.4$ to fuzzy set *large*, the value of A in e_2 has membership degrees $\mu_{\text{small}}(e_1) = 0.5$ to small and $\mu_{\text{large}}(e_2) = 0.5$ to large. We notice that in this illustration, the sum of the membership degrees of the attribute's value to the two fuzzy sets of attribute A is actually 1.
- After splitting the training set w.r.t. A, the membership degrees of e_1, e_2 will be updated as illustrated in Table 4: e.g. $\chi_1^{N|\text{small}} = \top_{\min}(\chi_1^N, \mu_{\text{small}}(e_1)) = \min(0.6, 0.6) = 0.6$. If we interpret the membership degrees to the fuzzy example set as case weights we can sum the weights for the subsets to obtain the weights for the whole set. In this way we obtain the case weights $\chi_1^N = 0.6 + 0.4 = 1.0$ for e_1 and $\chi_2^N = 0.5 + 0.5 = 1.0$ for e_2. The frequency distribution of the classes w.r.t. these case weights is $p_{C_1} : p_{C_2} = 1 : 1$.

Here occurs the same situation as in Example 1, and due to the same reason the conclusion – $\text{Gain}(\chi^N, A) = I(\chi^N) - I(\chi^N|A) < 0$ is possible – can be drawn. In Example 2 we obtain $\text{Gain}(\chi^N, A) = 0.971 - 0.993 = -0.022$.

3.5 Extended information measures

In Sect. 3.4 we have shown that negative information gain as well as negative gain ratio can occur in FDT induction (e.g. in the version developed in Janikow 1998) if we use the entropy $I(\chi^N)$ (e.g. in Table 1 of Example 1) computed from the membership degrees of the examples (interpreted as case weights) in the current node, in which a test attribute is to be chosen, since the premiss of the classical information gain can be violated due to the fact, that the case weight of an example case in the whole fuzzy example set before and after a split can differ.

Table 4 Example 2 (membership degrees of examples after split)

After split	C_1	C_2
	e_1	e_2
Small	$\top_{\min}(0.6, 0.6)$=0.6	$\top_{\min}(0.9, 0.5)$=0.5
Large	$\top_{\min}(0.6, 0.4)$=0.4	$\top_{\min}(0.9, 0.5)$=0.5
$\hat{\chi}_j^N$	1.0	1.0
$\hat{Z}^N$	1.0	1.0

A negative information gain, although it has no real meaning, can still yield a correct ranking of the candidate test attributes. But if information gain ratio is used, a negative value for the information gain can produce an inappropriate answer. To see this, let us consider a simple example: suppose for the current tree node N we have two candidate attributes A and B with information gain $\text{Gain}(\chi^N, A) > \text{Gain}(\chi^N, B)$ and split information $\text{SplitI}(\chi^N, A) \gg \text{SplitI}(\chi^N, B)$, where χ^N is the fuzzy example set (a fuzzy set over a reference set E of examples) in N, which acts actually as the training set.

1. In classical decision trees it is always $\text{Gain}(\chi^N, A) > \text{Gain}(\chi^N, B) \geq 0$, and then it may be that

$$0 \leq \frac{\text{Gain}(\chi^N, A)}{\text{SplitI}(\chi^N, A)} < \frac{\text{Gain}(\chi^N, B)}{\text{SplitI}(\chi^N, B)}. \tag{3}$$

   This is desired, because it reduces the well-known bias of information gain towards many-valued attributes as described above.
2. In the fuzzy domain, however, we can also have the situation $\text{Gain}(\chi^N, A) > 0 > \text{Gain}(\chi^N, B)$. In such a case, attribute A is always favored by the information gain ratio, because

$$\frac{\text{Gain}(\chi^N, A)}{\text{SplitI}(\chi^N, A)} > 0 > \frac{\text{Gain}(\chi^N, B)}{\text{SplitI}(\chi^N, B)}, \tag{4}$$

   independent of the relationship between SplitI (χ^N, A) and $\text{SplitI}(\chi^N, B)$, and this contradicts our intuition.

To make information gain ratio applicable as a selection measure, we suggest a different way of computing the information measure in the fuzzy domain. Using the entropy $I(\hat{\chi}^N)$ of the examples (derived from the case weights $\hat{\chi}^N$ as computed above, e.g. in Table 2 of Example 1), which implicitly includes the information of the test attribute, we can guarantee that the information measure is non-negative. In the following we explain our extensions in a formal way.

Suppose we have a reference set $E = \{e_1, \ldots, e_s\}$ coming from m different classes $C = \{C_j | 1 \leq j \leq m\}$ and χ^N is the fuzzy example set in node N. A is a candidate test attribute that has a fuzzy partition with $|D|$ fuzzy

Table 5 (Fuzzy) contingency table for node N with attribute A as the test candidate

$N(A)$	C_1	C_2	...	C_m	$ASum$
a_1	$Z_{C_1}^{N\|a_1}$	$Z_{C_2}^{N\|a_1}$	...	$Z_{C_m}^{N\|a_1}$	$Z^{N\|a_1}$
a_2	$Z_{C_1}^{N\|a_2}$	$Z_{C_2}^{N\|a_2}$	...	$Z_{C_m}^{N\|a_2}$	$Z^{N\|a_2}$
...	...	...	...	...	...
a_p	$Z_{C_1}^{N\|a_p}$	$Z_{C_2}^{N\|a_p}$	...	$Z_{C_m}^{N\|a_p}$	$Z^{N\|a_p}$
$CSum$	$Z_{C_1}^{N}$	$Z_{C_2}^{N}$	...	$Z_{C_m}^{N}$	Z^{N}

sets $\{a_p|1 \le p \le |D|\}$. u_k^A is the value of attribute A in example e_k, $1 \le k \le s$.

To determine the best test attribute, we create a (fuzzy) contingency table (see Table 5) for each candidate A in node N, from which we can compute the information measure for A. In Table 5, we have:

1. $Z_{C_j}^{N|a_p}$ – the counter for the examples that belong to fuzzy set a_p and class C_j
2. $Z^{N|a_p}$ in column "ASum" – the counter for all examples in child node $N|a_p$ (i.e. the sum of the examples' membership degrees to the fuzzy example subset for fuzzy set a_p): $Z^{N|a_p} = \sum_{j=1}^{m} Z_{C_j}^{N|a_p}$.
3. $Z_{C_j}^{N}$ in row "CSum" – the counter for the examples which belong to class C_j.
4. Z^N – the counter for the entire set of examples, we notice that $Z^N = \sum_{p=1}^{|D|} Z^{N|a_p} = \sum_{j=1}^{m} Z_{C_j}^{N}$.

Figure 6 presents the algorithm for estimating the information measure of the candidates. The basic computation is similar to that in classical trees: to determine information gain or gain ratio, the generalized entropy in node N as well as the average entropy of all child nodes are computed. However, the significant difference is to compute the membership degree of each example e_k to the subset in each child node corresponding to fuzzy set a_p — $\chi_k^{N|a_p}$, where we first have to get the membership degree of attribute value u_k^A in each example e_k to a_p — $\mu_{a_p}(u_k^A)$ (line 8), then $\chi_k^{N|a_p} = \top_1(\mu_{a_p}(u_k^A), \chi_k^N)$ (line 9) is obtained. Afterwards, we compute the different example counters — $Z_{C_j}^{N|a_p}$ (line 10), $Z^{N|a_p}$ (line 14), $Z_{C_j}^{N}$ (line 12), Z^N (line 15), from which various class frequency distributions and entropies can be determined: the entropy in each fuzzy example subset $I(\chi^{N|a_p})$ (line 18), the weighted sum of entropies $I(\chi^N|A)$ (line 19) of the children, and the split information $\mathrm{SplitI}(\chi^N, A)$ (line 21) of A. The most crucial point here is, to guarantee non-negative information gain. We propose to use the entropy $I(\hat{\chi}^N)$ based on fuzzy example set $\hat{\chi}^N$ implic-

Algorithm 2.
Compute the information measure in node N (*candidate attribute* A, *training set* $\{\chi_k^N\}$)

/ At first, we initialize the counters* $Z_{C_j}^{N|a_p}$, $Z_{C_j}^{N}$, $Z^{N|a_p}$, Z^N **/*

create the fuzzy contingency table for A and initialize the counters in it:

1. $Z_{C_j}^{N|a_p} = 0$ (for each j, p with $1 \le j \le m, 1 \le p \le |D|$)
2. $Z_{C_j}^{N} = 0$ (for each j with $1 \le j \le m$)
3. $Z^{N|a_p} = 0$ (for each p with $1 \le p \le |D|$)
4. $Z^N = 0$

/ calculate the counters:* $Z_{C_j}^{N|a_p}$, $Z_{C_j}^{N}$, $Z^{N|a_p}$, Z^N **/*

5. for each class C_j, $1 \le j \le m$:
6. for each fuzzy set a_p of A, $1 \le p \le |D|$:
7. for each example e_k, $1 \le k \le s$:
8. calculate the membership degree of attribute value u_k^A to fuzzy set a_p as $\mu_{a_p}(u_k^A)$
9. calculate the membership degree of example e_k to the fuzzy example subset for fuzzy set a_p: $\chi_k^{N|a_p} = \top_1(\mu_{a_p}(u_k^A), \chi_k^N)$
10. compute the counter for the examples belonging to fuzzy set a_p and class C_j: $Z_{C_j}^{N|a_p}$ += $\top_2(\chi_k^{N|a_p}, y_k^j)$
11. end k
12. get the counter for the examples that belong to class C_j: $Z_{C_j}^{N}$ += $Z_{C_j}^{N|a_p}$
13. end p
14. get the counter for the examples in child node $N|a_p$: $Z^{N|a_p}$ += $Z_{C_j}^{N|a_p}$
15. get the counter for the entire examples after split: Z^N += $Z_{C_j}^{N}$
16. end j

/ now we have got the counters* $Z_{C_j}^{N|a_p}$, $Z_{C_j}^{N}$, $Z^{N|a_p}$, Z^N, *and can use them to compute the information measure straightforward */*

17. from row "CSum"in Table 5, calculate the class frequency distribution and the entropy of the entire examples in N: $I(\hat{\chi}^N) = -\sum_{j=1}^{m} \left(\frac{Z_{C_j}^N}{Z^N}\right) \log_2 \left(\frac{Z_{C_j}^N}{Z^N}\right)$
18. for each child node $N|a_p$, from Table 5 we compute line-by-line the entropy of each fuzzy example subset in it as: $I(\chi^{N|a_p}) = -\sum_{j=1}^{m} \left(\frac{Z_{C_j}^{N|a_p}}{Z^{N|a_p}}\right) \log_2 \left(\frac{Z_{C_j}^{N|a_p}}{Z^{N|a_p}}\right)$
19. calculate the weighted sum of entropies $I(\chi^N|A)$ of the children as: $I(\chi^N|A) = \sum_{p=1}^{|D|} \frac{Z^{N|a_p}}{Z^N} I(\chi^{N|a_p})$.
20. then the information gain, using A as test attribute, is: $\widehat{\mathrm{Gain}}(\chi^N, A) = I(\hat{\chi}^N) - I(\chi^N|A)$.

/ if information gain ratio is used, we need two more steps: */*

21. the split information $\mathrm{SplitI}(\chi^N, A)$ of attribute A is computed from $Z^{N|a_p}$: $\mathrm{SplitI}(\chi^N, A) = -\sum_{p=1}^{|D|} \left(\frac{Z^{N|a_p}}{Z^N}\right) \log_2 \left(\frac{Z^{N|a_p}}{Z^N}\right)$
22. the information gain ratio of attribute A is then computed as $\widehat{\mathrm{GainR}}(\chi^N, A) = \widehat{\mathrm{Gain}}(\chi^N, A) \,/\, \mathrm{SplitI}(\chi^N, A)$.

Fig. 6 calculate the information measure

itly including the information of test attribute A, instead of fuzzy example set χ^N in node N, for χ^N and $\hat{\chi}^N$ can be different due to fuzzy logic. Since the counter $Z^N_{C_j}$ and Z^N used for entropy $I(\hat{\chi}^N)$ are computed from $\chi_k^{N|a_p}$, this means $\hat{\chi}_k^N = \sum_{p=1}^{|D|} \chi_k^{N|a_p}$ $(1 \leq k \leq s)$, more clearly, $I(\hat{\chi}^N)$ is calculated from the same case weights as $I(\chi^N|A)$. Therefore $\widehat{\text{Gain}}(\chi^N, A)$ is guaranteed to be **non-negative.** As consequent, non-negative gain ratio $\widehat{\text{GainR}}(\chi^N, A)$ is also ensured.

Remark 1 information gain ratio is used in C4.5 Quinlan (1993) to select the test attribute in order to reduce the natural bias of information gain, i.e., the fact that it favors attributes with many values (which may lead to a model of low predictive power). In FDT induction, fuzzy partitions are created for all attributes before the tree induction. To keep the tree simple, usually each partition possesses as few fuzzy sets as possible. Since the outgoing branches are labelled with fuzzy sets instead of crisp values, the problem mentioned above is mitigated, because continuous values are mapped to few fuzzy sets and thus the problem of many values is less severe. Therefore, the effect of using information gain ratio in FDT may not be so obvious as in classical decision trees.

In following, we give an example to show the difference in choice of attributes between using information gain and gain ratio with and without the proposed extension. Given the example reference set E shown in Example 1 (see Table 1). Assume there are two candidate attributes A, B. Attribute A has two fuzzy sets small and large, attribute B has four fuzzy sets *s1*, *s2*, *s3* and *s4*. The split according to A is illustrated in Table 2 (in Example 1), and the split according to B in Table 6. When we use information gain as the selection criterium, based on the original generalized version (e.g. Janikow (1998)), we have

$$\text{Gain}(\chi^N, A) = 0.9710 - 0.9793 = -0.0083$$
$$\text{Gain}(\chi^N, B) = 0.9710 - 0.9316 = 0.0394.$$

Table 6 Example 3 (membership degrees of examples after split)

Split according to B	C_1		C_2		
	e_1	e_2	e_3	e_4	e_5
s1	0.3	0	0.4	0	0.6
s2	0.7	0	0.6	0	0
s3	0	0.6	0	0.8	0
s4	0	0.4	0	0.2	0.4
$\hat{\chi}_j^N$	1.0	1.0	1.0	1.0	1.0
$\hat{Z}^N$	2.0			3.0	

$\text{Gain}(\chi^N, B)$ is larger than $\text{Gain}(\chi^N, A)$, B is thus more favored than A. When we use the modified measure based on our amendments:

$$\widehat{\text{Gain}}(\chi^N, A) = 1 - 0.9793 = 0.0207$$
$$\widehat{\text{Gain}}(\chi^N, B) = 0.9710 - 0.9316 = 0.0394$$

in case of B, we notice $\forall 1 \leq k \leq 5 : \hat{\chi}_k^N = \chi_k^N$, hence information gain with respect to B based on our amendments has not changed, B is still chosen. The split information of attribute A and B are:

$$\text{SplitI}(\chi^N, A) = 0.9799$$
$$\text{SplitI}(\chi^N, B) = 1.9892.$$

Considering gain ratio without the extensions:

$$\text{GainR}(\chi^N, A) = -0.0083/0.9799 = -0.0085$$
$$\text{GainR}(\chi^N, B) = 0.0394/1.9892 = 0.0198.$$

A is still less valued merely due to the negative value, i.e., in such a case the split information of the two attributes actually do not have the expected effect. If we use the gain ratio based on our extension, we will have

$$\widehat{\text{GainR}}(\chi^N, A) = 0.0207/0.9799 = 0.0211$$
$$\widehat{\text{GainR}}(\chi^N, B) = 0.0394/1.9892 = 0.0198$$

obviously, now the split information will be correctly taken into account and A is favored. Therefore with the proposed extension, gain ratio is made applicable in the fuzzy domain according to the motivation that it was proposed with.

3.6 Missing value handling

Data collected from real world often contain missing values. To handle such data we extend the learning algorithm, so that deleting examples with missing values from the training data set is not necessary anymore.

The first question to be answered is how to assign the examples with missing values of the test attribute to the outgoing branches of a tree node. In this paper a popular method from classical decision trees is adopted: an example e_k is distributed evenly to all children if the value u_k^i for test attribute A_i is unknown. It means that to each fuzzy set (branch) a_p^i of A_i example e_k is assigned with membership degree $\mu_{a_p^i}(u_k^i)$:

$$\mu_{a_p^i}(u_k^i) = \frac{1}{|D_i|} \quad \text{if } u_k^i \text{ is unknown}, \tag{5}$$

where $|D_i|$ is the number of the fuzzy sets of A_i.

As we know, the information gain can be interpreted as "the information gained about the classes by ascertaining the value of the test attribute", a test of

an example with a missing value for the test attribute, can obviously provide no information about the class membership of this example. Therefore, the assessment of candidate attributes has to be modified accordingly, so that attributes with missing values are penalized.

Suppose we are given a reference set E having missing values for attribute A_i. Then the calculation of the information gain for candidate attribute A_i can, as suggested in Quinlan (1993), be modified as following:

$$\begin{aligned}&\widehat{\text{Gain}}(\chi^N, A_i)\\&\quad = \text{frequency of examples with known}\\&\qquad A_i \cdot (I(\hat{\chi}^N) - I(\chi^N|A_i))\\&\qquad +\text{frequency of examples with unknown}\\&\qquad A_i \cdot 0\\&\quad = \alpha \cdot (I(\hat{\chi}^N) - I(\chi^N|A_i)),\ \text{where } \alpha = \frac{Z^{N|A_i \text{ known}}}{Z^N}.\end{aligned} \tag{6}$$

Due to the factor α, which is computed from the counter for examples without missing values of A_i "$Z^{N|A_i \text{ known}}$" and the counter for the entire examples "Z^N", the real information gain is only dependent on those examples with known values for the test attribute.

The information gain ratio can be amended in a similar way:

$$\widehat{\text{GainR}}\left(\chi^N, A_i\right) = \alpha \cdot \frac{I\left(\hat{\chi}^N\right) - I\left(\chi^N|A_i\right)}{\text{SplitI}\left(\chi^N, A_i\right)}. \tag{7}$$

Since the split information $\text{SplitI}\left(\chi^N, A_i\right)$ is the entropy of the frequency distribution over the values of attribute A_i, the split information is increased artificially by evenly splitting the examples with missing values, and the information gain ratio is decreased accordingly (since $\text{SplitI}(\chi^N, A_i)$ appears in the denominator). This effect is desired, because an attribute having missing values should be penalized. Since the increased split information already penalizes the measure, one may consider making the use of the factor α (see above) optional. That is, it is added only when a user explicitly requests it.

3.7 Fuzzy rule base

An important goal of this work is to generate a comprehensible classification model, here a fuzzy rule base, which can be generated from the fuzzy decision tree, that has been learned from data as described above.

3.7.1 Fuzzy rule learning

There are different approaches to learn a fuzzy system, the three most important categories are: cluster-, hyperbox- and structure-oriented learning. The first two approaches generate fuzzy rules and fuzzy sets at the same time, while the last one defines the granularity of the data space in advance, i.e., it creates initial partitions for all variables, then produces the rule base from this structured data space.

Cluster-oriented rule learning is an unsupervised learning method. With fuzzy cluster analysis (Bezdek et al. 1998), the training data are grouped into clusters, then the fuzzy rules are generated by transforming each cluster into a rule. Fuzzy sets are obtained by projecting the clusters onto the domain of each variable.

Hyperbox-oriented learning is a supervised learning approach, where the training data are covered by (eventually overlapping) hyperboxes, so that the dependency between output and input variables is described by fuzzy graphs (Berthold and Huber 1999). The fuzzy rules are produced from the hyperboxes and fuzzy sets by projecting the hyperboxes onto individual dimensions.

However, these two methods have some drawbacks in common:

1. Each fuzzy rule uses individual fuzzy sets, thus the rule base is difficult to interpret.
2. The fuzzy sets generated by the projection are hard to interpret linguistically.
3. They cannot cope with missing values.

Cluster-oriented approaches have besides the restriction that a suitable number of clusters must be determined by repeating the cluster analysis with an increasing number of clusters until some validity measure assumes a local optimum.

Structure-oriented learning suggested by Wang and Mendel (1992) avoids the drawbacks mentioned above. By providing initial fuzzy sets for all variables, the data space is structured by a multidimensional fuzzy grid, and the rules are created by selecting those cells containing data. Fuzzy decision trees can be viewed as a structure-oriented fuzzy rule learning method with concurrent structure optimization, where the data space is partitioned in a data-driven manner and the partition is represented as a tree. The fuzzy rules are generated by transforming each path to a leaf of the tree into a rule. The antecedent of each rule consists of the fuzzy sets along that path leading to the leaf and the consequent of each states membership degrees for the classes which can be read directly from that leaf. Fuzzy decision tree learning has the following advantages over cluster- and hyperbox-oriented learning: it is able to deal with missing values, allows an unambiguous representation of linguistic terms and provides better interpretability of the resulted rule base.

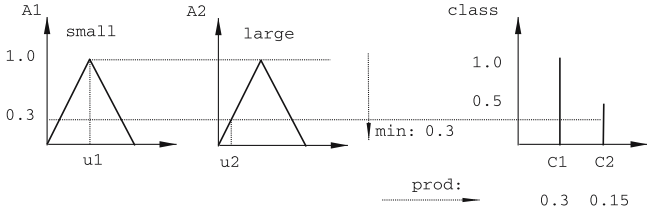

Fig. 7 A fuzzy rule example

3.7.2 Classifying data

To classify data, the fuzzy rule base generated from the fuzzy decision tree works in basically the same manner as standard fuzzy systems.

1. A fuzzy rule is $R_k = (AR_k, CR_k)$ in the form of: "If A_1 is μ_k^1, A_2 is $\mu_k^2, \ldots,$ A_n is μ_k^n then $C_1 : w_k^1$, $C_2 : w_k^2, \ldots,$ $C_m : w_k^m$", where μ_k^i $(1 \le i \le n)$ is a fuzzy set of input variable A_i. $AR_k = (a_k^1, \ldots, a_k^n)$ is the antecedent. $CR_k = (w_k^1, w_k^2, \ldots, w_k^m)$ is the consequent, which contains all possible classes with corresponding membership degrees $w_k^j (1 \le j \le m)$, and is described by a set of fuzzy singletons $(C_m : w_m)$.
2. $\top_1$ (t-norms) conjunctively combines the membership degrees of input values u^i $(1 \le i \le n)$ to all fuzzy sets in the antecedent, to determine the fulfillment degree τ_k of rule R_k: $\tau_k = \top_1 \left\{\mu_k^1(u^1), \mu_k^2(u^2), \ldots, \mu_k^n(u^n)\right\}$.
3. The inference operation $\top_2$ (t-norms) is used to compute the conclusion of a rule, i.e., the consequent of rule R_k can be computed as $y_k^j = \top_2(\tau_k, w_k^j)$. The output of the fuzzy system for an input example is a vector of all classes and is described as $\mathbf{y} = (y^1, \ldots, y^m)$.
4. The defuzzification method "defuzz" determines the crisp output value, here two options *MAX* and *MEAN* are given.

To classify an example, the decision for it has to be made by the entire set of fuzzy rules. Figure 7 gives a simple illustration to clarify how one fuzzy rule classifies a data example. Given is a data example with two input variables A_1 and A_2, whose crisp values u_1 and u_2 have the membership degrees of 1.0 and 0.3 to fuzzy sets small and large respective ly. Here we use $\top_1 = \top_{\min}$ and $\top_2 = \top_{\text{Prod}}$ merely for the illustration: we will get a fulfilment degree of the rule of 0.3 by using $\top_1 = \min(1.0, 0.3)$. The consequent consists of two classes with membership degrees 1.0 and 0.5 respectively. With $\top_2 = \top_{\text{Prod}}$ the consequent of the rule classifies this data example as class C_1 with $y^1 = \top_{\text{Prod}}(1.0, 0.3) = 0.3$ and class C_2 with $y^2 = \top_{\text{Prod}}(0.5, 0.3) = 0.15$, it means this example is classified as class C_1 if only this rule is considered.

After all rules have made their individual decisions for a data example, in order to determine a class for this data, we have to convert these decisions into one crisp output by the defuzzification method. Here we define two variants: *MAX* and *MEAN*.

Given a fuzzy-classifier with r rules $R = \{R_1, \ldots, R_r\}$ that can predict three classes C_1, C_2, C_3. The output y_k^j of each rule R_k (calculated in the same way as in Fig. 7) is shown in Table 7.

The output vector of this classifies – in this example: $\mathbf{y} = (y^1, y^2, y^3)$ – can be calculated by using either of these two variants of defuzzification. The method *MAX* determines the output $\mathbf{y}$ by finding the maximal value for each class (i.e., in each column):

$$y^j = \text{defuzz}_{\text{MAX}}(y_k^j) = \max_{1 \le k \le r} \{y_k^j\}, \quad 1 \le j \le 3.$$

With *MEAN* all values will first be weighted (by the fulfillment degree of the corresponding rule) accumulated column by column for each class, then these sums will be divided by the sum of the fulfillment degrees of all rules:

$$y^j = \text{defuzz}_{\text{MEAN}}(y_k^j) = \frac{\sum_{k=1}^r (\tau_k \cdot y_k^j)}{\sum_{k=1}^r (\tau_k)}, \quad 1 \le j \le 3.$$

If a fuzzy or continuous result is desired, this output vector can be put to use directly. If a crisp decision has to be made, the Winner-Take-All-Interpretation can be applied, i.e., all components of the output vector should be compared and the class corresponding the largest value will be put out: if e is a data to be classified, then $\text{class}(e) = \text{argmax}_{j:1 \le j \le 3}\{y^j\}$. In case all components of the output vector are equal, i.e., $y^1 = y^2 = y^3$, "not classified" will be yielded as the result.

Table 7 Outputs of the fuzzy rules

	C_1	C_2	C_3
R_1	y_1^1	y_1^2	y_1^3
$\ldots$	$\ldots$	$\ldots$	$\ldots$
R_k	y_k^1	y_k^2	y_k^3
$\ldots$	$\ldots$	$\ldots$	$\ldots$
R_r	y_r^1	y_r^2	y_r^3

3.7.3 Pruning strategies

A simpler model or a model with better predictive power cannot be produced by just rewriting the fuzzy decision tree into the form of fuzzy rules. To achieve this an optimization of the rule base is necessary.

We optimize the rule base by rule pruning, where three heuristic strategies adapted from Nauck et al. (1999) are used:

1. Pruning by information measure: the attribute having the smallest influence on the output should be deleted.
2. Pruning by redundancy: the linguistic term, which yields the minimal membership degree in a rule in the least number of cases, should be deleted.
3. Pruning by classification frequency: the rule, which yields the maximal fulfillment degree in the least number of cases, should be deleted.

These strategies are effective in both reducing the number of rules and increasing generalization ability. Since the comprehensibility of a fuzzy system can be defined by the number of the rules, the number of attributes used in a rule and the number of fuzzy sets per attribute, the heuristics used in the strategies above are plausible.

Before going to more details, we first describe how to evaluate the performance of a model. The performance of a classification model can be indicated by some error measures. Given a data set $E = \{e_1, \ldots, e_s\}$ of examples $e_k = (\mathbf{u}_k, \mathbf{v}_k)$, $1 \leq k \leq s$. $\mathbf{u}_k$ is the input vector and $\mathbf{v}_k \in [0,1]^m$ the target output vector of e_k. $\mathbf{y}_k \in [0,1]^m$ is the real output of the classifier and computed as descried in Sect. 3.7.2. Two measures – error and misclassified – are defined as follows:

1. error measures: how crisp the classification is.
   The error made on a single example is defined as the squared difference between the target output and the real output $\sum_{j=1}^{m}(v_k^j - y_k^j)^2$. The measure error of the classifier over E is the sum of the errors of all examples:

$$\text{error} = \sum_{k=1}^{s}\sum_{j=1}^{m}(v_k^j - y_k^j)^2. \tag{8}$$

2. misclassified measures: the number of misclassified examples.
   The measure misclassified uses the crisp output of the classifier instead of the output vector. The target class of e_k is $\text{class}(e_k) = \text{argmax}_{j:1\leq j\leq m}\{v_k^j\}$, and the real crisp classification made by the classifier is likewise determined $\text{class}'(e_k) = \text{argmax}_{j:1\leq j\leq m}\{y_k^j\}$. The error made on an example is either 1, if the real output class coincides with the target class; or 0 otherwise. Thus the measure "misclassified" over data set E is:

$$\text{misclassified} = \sum_{k=1}^{s}\text{merr}_k,$$

where

$$\text{merr}_k = \begin{cases} 1, & \text{falls } \text{class}'(e_k) = \text{class}(e_k) \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

These two error measures will not only be used as the error estimation of classifiers, but also as stop criteria for the optimization of the rule base.

The three strategies described above are called successively by the pruning algorithm. At each pruning step only one attribute or one fuzzy set or one rule is deleted from the system. Before each step the current rule base is to be saved. Once the execution of one strategy is finished, we should check, whether this pruning step improved the rule base successfully. If the rule base simplified by this step has a better performance than the original one, the changed rule base is kept and then is to be pruned further with the same strategy. Otherwise the rule base should be reset with the previously saved version, and the pruning will be continued with the next strategy. The pruning process can work automatically without any user interactions.

There are four check modes available for the performance check of the rule base after each pruning step:

1. error_mode: check, whether the value of error is smaller after pruning.
2. misclassified_mode: check, whether the value of misclassified is smaller after pruning.
3. error_ and misclassified_mode: check, whether the values of both measures decreased after pruning.
4. error_ or misclassified_mode: check, whether one value of these two measures decreased after pruning.

To reduce runtime, one pruning strategy will be iterated until it fails to improve the rule base, then the next strategy is selected. The algorithm gives the user the opportunity to choose the "exhaustive pruning". In this mode, after a failed pruning step the same strategy will be carried on with the next parameter (this can be an attribute, a linguistic term or a rule) it recommends for pruning, till all possible parameters are used, then

Table 8 Test data

Data	Size	Attributes	Classes	Missing value
Iris	150	4	3	No
Glass	214	10 (incl. Id)	7	No
Thyroid	215	5	3	No
WBC	699	10 (incl. Id)	2	Yes
Pima	768	8	2	No

Table 9 Tenfold cross validation

Model		Iris	Glass	Thyroid	WBC	Pima
FDT	$\bar{\varepsilon}$	4.67%	34.29%	**3.33%**	2.79%	31.32%
(1)	n	3	11	5	12	2
FDT	$\bar{\varepsilon}$	5.33%	31.90%	7.62%	2.64%	**18.82%**
(2)	n	3	33	10	17	40
C4.5	$\bar{\varepsilon}$	4.01%	33.54%	7.03%	4.83%	23.3%
	n	4	14	7	8	8
NEFCLASS	$\bar{\varepsilon}$	**3.33%**	32.19%	11.60%	**2.35%**	25.89%
	n	3	14	6	21	14
NN	$\bar{\varepsilon}$	6.25%	**31.27%**	3.54%	5.05%	24.67%

switches to the next strategy. With "exhaustive pruning" the rule base will be pruned most thoroughly, as consequence a longer runtime is necessary.

4 Experiments

In this section, we report some results obtained from experiments run with the program FDT,[3] which was written by the first author of this paper, the well-known decision tree learner C4.5 (Release 8)[4] Quinlan (1993), a neural network training program Borgelt, and NEFCLASS Nauck and Nauck, which can generate a fuzzy rule based classifier by coupling neural networks with fuzzy systems. We compare the models generated by these programs w.r.t. precision, complexity, and the ability of dealing with missing values. For the tests we used five data sets from the UC Irvine Machine Learning Reposity Blake and Merz. Table 8 shows general information about these data sets.

All experiments were run with tenfold cross validation. C4.5 was run with the standard configuration. In NEFCLASS for each attribute a fuzzy partition with three evenly distributed fuzzy sets was created. Fuzzy sets were also optimized during the rule pruning. The neural network program trained a multilayer perceptron (MLP) with one hidden layer (3 neurons) for 1,000 epochs.

Since we tried to generate comprehensible classification models, a trade-off between precision and complexity should be found. With this concern in mind, in FDT a threshold of 0.05 for the information measure was chosen. That is, a test is created only if the chosen test attribute yields an information value higher than 0.05.

4.1 Precision and complexity

Table 9 shows the average error rate $\bar{\varepsilon}$, as well as the number of rules n of the resulting pruned classifiers. The best error rate of the models is printed in bold in the table.

In these experiments, FDT was run with two different initial partitioning of the attributes – the automatic (labelled as FDT (1)) and the individual partitioning (labelled as FDT (2)) mentioned above. With the individual partitioning each attribute was partitioned with three fuzzy sets, which were evenly distributed over the attribute's domain, while with automatic partitioning the number of fuzzy sets was determined by the program.

If we consider only the precision of the models, it is very difficult to say which method is the best one, since each method produces the best result at least once. C4.5 never yields the worst error rate. FDT (1) gives the highest precision (3.33%) for the *thyroid* data and at the same time a very small rule base (5 rules). For the WBC data NEFCLASS achieves the best performance of 2.35% with as many as 21 rules, while FDT (1) provides performance only slightly worse (2.79%), for which it needs only about half the rules (12). The neural network fares worst for the wbc, which is probably due to the fact that an MLP with three hidden neurons (as used here) is comparable in power to about three rules. With so few rules no good performance can be expected for the wbc data.

For the pima data the worst classification rate is provided by FDT with the automatic partitioning (however, with only 2 rules). The reason is that the partitioning algorithm created for only 2 of the 8 attributes two fuzzy sets and only one fuzzy set for each of the rest. Therefore the potential number of rules is only four, with which no learning algorithm can do much. In a comparison with the best result of FDT (with individual partitioning, which required as many as 40 rules), we noticed that the attributes of the pima data have a relatively strong interrelationship. Therefore the data can be predicted better only by combining several attributes. A finer granularity, which was achieved by FDT with the individual partitioning, enhanced the probabil-

[3] We used the rule base generated by FDT for the experiments.

[4] The learning result of C4.5 can be both a tree or a rule base. Here we used the generated rule base for the experiments.

Table 10 Learning from data with missing values

Data		5%			10%		
		FDT	C4.5	Nefclass	FDT	C4.5	Nefclass
Iris	$\bar{\varepsilon}$	**4.67%**	8.01%	5.33%	10.67%	12.00%	**6.67%**
	n	4	5	3	3	3	3
Glass	$\bar{\varepsilon}$	39.52%	**34.61%**	37.19%	**29.04%**	40.25%	39.18%
	n	13	11	18	21	10	23
Thyroid	$\bar{\varepsilon}$	**3.81%**	8.34%	33.92%	**8.57%**	10.24%	18.53%
	n	6	7	3	4	6	5
WBC	$\bar{\varepsilon}$	5.51%	**4.86%**	4.87%	7.68%	**5.28%**	5.58%
	n	23	11	32	20	12	39
Pima	$\bar{\varepsilon}$	31.45%	26.71%	**22.66%**	35.00%	**27.23%**	27.59%
	n	2	8	21	2	6	25

ity of a combination of attributes, and thus led to a better performance.

The same partitioning like in FDT (2) was also used in NEFCLASS. For the *pima* data NEFCLASS provided a slightly lower precision, but with less than half the rules. We assume that the reason is that the fuzzy sets of NEFCLASS were trained during the learning and pruning phase, so they probably fit the data better. In contrast to this the fuzzy sets used in FDT (2) were created once at the beginning and did not change anymore.

If we compare the two groups of results yielded by FDT – taking not only the precision but also the complexity of the classifiers into account – we conclude that the learning process creates better classifiers if it works with automatic instead of individual partitioning. In particular, the number of rules of the first variant is often clearly less than that of the latter. Presumably the reason is that in the first variant the class information is taken into account, whereas it is neglected in the latter.

4.2 Tests on imperfect data

The experiments on the data with missing values, which were generated by randomly deleting values from each data set, demonstrate how well different learning methods can cope with imperfect data. In these tests FDT was only run with automatic partitioning.

Two columns of Table 10[5] show the results for data sets with 5 and 10% missing values, respectively. The best results are printed in bold face. As expected, the performance of all methods decreased with the increased portion of missing values. FDT provided for the thyroid data (both 5% and 10% missing values) the best result, as well as for the iris data (5%) and the glass data (10%). However, it is impossible to single out a method that is consistently superior to the others. The properties of the data seem to have stronger influence than the portion of missing values.

In C4.5 the threshold values for tests of continuous attributes are determined dynamically and locally in the nodes; in NEFCLASS, although all attributes are partitioned with fuzzy sets before learning, the fuzzy sets are still optimized afterwards. In contrast to this the fuzzy sets used in FDT are not changed anymore after creation. The lack of such dynamic fitting may be a disadvantage of the resulting fuzzy decision tree.

5 Surface quality control – a real world application

In this section, we present a real world application for the quality control of car body panels with the goal of building an automatic estimation method based on classification techniques. At first, we give some background of the problem.

Today, car manufacturers of the upperclass and premium market segments differentiate their products from their competitors among other things by a perfect appearance of the painted car body. In general, the impression of a car is determined by an appealing design of its body, the color and gloss of its paint, and the manufacturing and assembly accuracy of the exterior body panels.

The geometric complexity of these panels makes them difficult to produce with metal forming technologies. Small surface form deviations like sink marks always exist. They result in inhomogeneous runs of light fringes on the highly reflective paint, which visibly disturb the perfect appearance of the car body. In order to eliminate or at least to minimize such surface defects the manufacturing process is optimized. Especially, it is imperative to control the quality of the parts directly in the first steps of the manufacturing process in the press shop.

[5] The neural network program does not appear in this table, since it cannot work with missing values.

Table 11 Surface form deviations

Class	Linguistic description	Frequency
Uneven surface	*several* sink marks in series or adjoined	62
Press mark	Local *smoothing* of (micro-)surface, heavier sink mark, deep depression preceeded by a low peak	53
Dent	*Rounded* damage inward, distinctive feature	51
Bulge	*Rounded* damage outward, distinctive feature, *relatively small* radius	47
Sink mark	*Slight* flat based depression inward	24
Uneven surface press mark	No decription, new category introduced by experts	19
Flat area	*Flat* plane on curved cumber surface	9
Uneven radius	Visible distortion of *radius* geometry	8

The current surface quality control procedure in the press shop is still done manually. Today, during series production an experienced worker checks the produced parts at the end of the press line in constant intervals, to detect form deviations, and derive their type and acceptance. The experts introduced a list of surface defects and characterizations, to that they conform more or less in their daily quality work. The surface form deviations are characterized by linguistic descriptions of their specific appearance, as shown in Table 11 for some common defects. The geometry of the defects is specified by vague terms and attributes.

However, the current method has several disadvantages. It is cumbersome, subjective, error-prone and time consuming, especially when analyzing the surface of large parts totally. Moreover the assessed parts are often lost for the manufacturing process. Therefore it would be desirable to have a more objective, non-contact, faster and automatic estimation method.

In Eichhorn et al. (2003) an approach based on the digitization of the exterior body panel surface with an optical measuring system (Fig. 8) is proposed. From the resulting point cloud, which contains the required geometric information of the surface defects, the authors try to characterize the form deviations by mathematical properties that are close to the subjective properties that the experts used in their linguistic descriptions. The approach has two major aspects: the quality specialists need information about the type of defect detected, and additionally they are interested in its severeness. Here, we focus on the first aspect. The characteristics of the described problem – its uncertainty, fuzziness and the use of expert knowledge – point to possible solutions in the field of soft-computing.

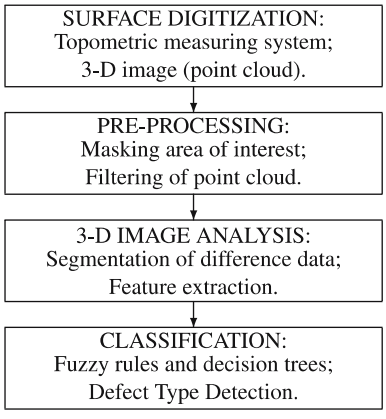

Fig. 8 Automatic quality assessment based on 3-D image processing

For details about the data acquisition and preprocessing refer to Eichhorn et al. (2003). The final database used for our analysis has a total number of 273 defects recorded by the experts, with 15 selected features after data preprocessing. The distribution of defect types is shown in Table 11. Obviously, the types are rather unbalanced, and the less frequent types occur very rarely.

Confidence into the system that predicts expert decisions is extremely important in responsible fields like quality control. The experts are more confident in a system, if its decisions are transparently and understandably given by rules or trees. Therefore, we compared the following approaches to classification: our fuzzy decision trees, decision trees and NEFCLASS. For the experiments we used fourfold cross validation. The database was split into four parts using stratified sampling to ensure that every split contains a similar distribution of defect types. Especially, this procedure ensures that each part contains at least one instance of each class. Experiments with the last two methods have been done in Eichhorn et al. (2003). To give a comparison with fuzzy decision tree approach, we briefly describe the results of them in the following.

Decision trees For the induction of the decision trees, several attribute selection measures have been tried, as described in Borgelt et al. (1996). Most of the measures

yield reasonable results. However, the Symmetric Gini Index maximized the tree accuracy over the training data set so it was employed as split criteria. For the pruning, confidence level pruning (Quinlan 1993) with a confidence of 50% was used. The classification accuracy is 95.85% on the training and 82.42% on the test data.

Nefclass NEFCLASS had some problems due to the high dimensionality of the dataset. In such cases, the structure-oriented approach by Wang and Mendel (1992) tends to produce too many, too specialized rules. Fuzzy set optimization gets unstable on such neuro-fuzzy networks, and as the pruning methods rely on an initial rule base, they might fail too. Therefore a selected subset of ten attributes was used. This made it easier to find good and general parameter settings for NEFCLASS. The best classification accuracy was 74.1% in average on the training sets and 75.44% on the test sets.

Fuzzy decision trees We have run the FDT program with automatic partitioning of attributes. A threshold of 0.06 for information gain ratio and exhaustive pruning was used, which yielded a reasonable trade-off between accuracy and complexity, the smallest rule base has only eight rules. The average classification accuracy is 75% on the training and 76.47% on the test data.

Since the data set we have was rather small and unbalanced, the classification was difficult. If only classification accuracy is of interest, decision trees would be the best choice. On the other hand, in the responsible field of quality control we also have to focus on the interpretability of the results. The experiments have shown that fuzzy rules obtained with NEFCLASS and FDT are in average less complex than the decision trees. Moreover, by selecting attributes with high information measure first, for fuzzy decision tree learning not all attributes are needed, thus FDT is more stable than NEFCLASS with respect to the dimension of attributes.

6 Conclusions

Intelligent data analysis is becoming increasingly important in businesses and industries. Believing that typical business users prefer softwares, which hide complexity from users and automate the data analysis process, over technology- or method-oriented ones, we presented a user-centered, automatic data analysis tool SPIDA. SPIDA is implemented in an client/server architecture. By using the fuzzy knowledge base, which includes fuzzy and non-fuzzy features of each analysis method, SPIDA can select and run data analysis methods, evaluate results and generate solutions automatically.

Within the framework of SPIDA, we combine fuzzy methods with classical decision trees in order to achieve comprehensible classifiers with the abilities of modelling vagueness. We studied the whole process from fuzzy tree learning, missing value handling to fuzzy rules generating and pruning. In particular, we investigated the information gain and information gain ratio based on the generalized Shannon Entropy for the test attribute selection and introduced amendments to avoid the negative values of them. We observe that fuzzy trees are normally larger than classical ones due to the membership degrees of the examples. To better control the complexity of the tree we suggested a threshold for the information measure. The presented heuristic pruning strategies are effective in both reducing the number of rules and increasing generalization ability. Compared to cluster- or hyperbox-oriented fuzzy rule learning methods, fuzzy decision tree learning has the following advantages: it is able to deal with missing values, allows an unambiguous representation of linguistic terms and provides better interpretability of the resulted rule base. Our experiments have shown, the approach proposed here often generates smaller and at the same time comparably good rule bases, hence our goal of obtaining comprehensible classifiers was reached.

We observe that the initial fuzzy partitioning of attributes strongly influences the quality of the final classifiers. Since the fuzzy sets are determined globally before tree induction and no training on them has been done during learning, the initial partitioning is especially important. Therefore, investigating other fuzzy partitioning techniques (e.g. the method proposed in Janikow and Fajfer (1999)) to enhance the quality of the initial partitioning will be our future work, as well as study of other measures for the selection of test attributes.

Acknowledgements The authors would like to thank Dr. Christian Borgelt for his constructive suggestions and helpful comments. Also many thanks to Mr. Christian Döring for his support on the real world application.

References

Berthold M, Huber KP (1999) Constructing fuzzy graphs from examples. Int. J. of Intelligent Data Analysis, 3(1)

Bezdek JC, Keller J, Krishnapuram R, Pal N (1998) Fuzzy models and algorithms for pattern recognition and image processing. Norwell Kluwer

Blake CL, Merz CJ UCI Repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html

Borgelt C `mlp` – multilayer perceptron training (computer software). http://fuzzy.cs.uni-magdeburg.de/˜borgelt/software.html#mlp

Borgelt C, Gebhardt J, Kruse R (1996) Concept for probabilistic and possibilistic induction of decision trees on real world data.

In: Proceedings 4th EUFIT 96, vol 3, pp 1556–1560, Aachen, Germany. Verlag Mainz

Borgelt C, Kruse R (2002) Graphical Models - Methods for Data Analysis and Mining. Wiley & Sons, Chichester

Bouchon-Meunier B, Marsala C (1999) Learning fuzzy decision rules. In: James C. Bezdek, Henri Prade, DuBois D (eds) Fuzzy sets in approximate reasoning and information systems, handbooks of fuzzy sets series, pp 279–304 Norwell, Kluwer

Bouchon-Meunier B, Marsala C, Ramdani M (1996) Inductive learning and fuzziness. MInt J Sci Technol, 2(4):289–298

Boyen XP, Wehenkel L (1995) Fuzzy decision tree inducion for power system security assessment. IFAC Symposium on control of power plants and power systems

Drobics M, Bodenhofer U (2002) Fuzzy modeling with decision trees. In: Proceedings of 2002 IEEE international conference on systems, man and cybernetics, Hammamet, Tunisia

Eichhorn A, Döring C, Klose A, Kruse R (2003) Classification of surface form deviations for quality analysis. In: Proceedings of the european symposium on intelligent technologied, hybrid systems and their implementation on smart adaptive systems(EUNITE 2003), pp 121–129, Oulu, Finnland

Janikow CZ (1998) Fuzzy decision trees: issues and methods. In: IEEE transactions on systems, vol 28, pp 1–14, Budapest Man and Cybernetics

Janikow CZ, Fajfer M (1999) Fuzzy Partitioning with FID3.1. In: Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society, pp 467–471 IEEE

Marsala C, Bouchon-Meunier B (2003) Choice of a method for the construction of fuzzy decision trees. In: Proceedings of the IEEE International Conference on fuzzy systems, St. Louis, USA, FuzzIEEE

Mitra S, Konwar KM, Pal SK (2002) Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: Generation and evaluation. In: IEEE transactions on systems, man and cyberneticsù-part c: applications and reviews, vol 32, no. 4

Nauck D, Nauck U Nefclass – neuro-fuzzy classification (computer software). http://fuzzy.cs.uni-magdeburg.de/˜nauck/software.html

Nauck D, Nauck U, Kruse R (1999) NEFCLASS for JAVA — New learning algorithmus. In: Proceedings of 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS99), New York IEEE, pp 472–476

Nauck D, Spott M, Azvine B (2003) SPIDA – a novel data analysis tool. BT Technol J 21(4):104–112

Nauck D, Spott M, Azvine B (2004) Fuzzy methods for automated intelligent data analysis. In: Proceedings of International Conference on fuzzy systems fuzzIEEE'2004, Budapest

Olaru C, Wehenkel L (2003) A complete fuzzy decision tree technique. Fuzzy Sets Syst, 138(2):221–254

Quinlan JR (1986) Induction of decision trees. Mach Learn, 1:81–106

Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufman

Wang LX, Mendel JM (1992) Generating fuzzy rules by learning from examples. In: IEEE Transactions on SMC, Man and Cybernetics 22(6):1414–1427

Yuan Y, Shaw M (1995) Induction of fuzzy decision trees. Fuzzy Sets syst 69:125–139