The Fuzzy Decision Tree Module in the Automatic Data Analysis Platform SPIDA

Xiaomeng Wang[1], Detlef D Nauck[2], Martin Spott[2], Rudolf Kruse[1]

[1] Faculty of Computer Science, University of Madgeburg

Universitaetsplatz 2, D-39106 Magdeburg, Germany

E-Mail: {xwang, rkruse}@iws.uni-magdeburg.de

[2] BT, Research and Venturing, Intelligent Systems Research Centre

Adastral Park, Orion Bldg. pp1/12

Ipswich IP5 3RE, UK

E-Mail: {detlef.nauck, martin.spott}@bt.com

June 16, 2004

Abstract

It is well known that classical trees lack the ability of modelling vagueness. By connecting fuzzy systems and classical decision trees, we try to achieve classifiers that can model vagueness and are comprehensible. We discuss the core problem of how to compute the information measure used in the induction of fuzzy trees and propose some improvements. In addition, we consider fuzzy rule bases derived from fuzzy decision trees and present some heuristic strategies to prune them. We also compare results of some of our experiments and compare our approach to other well-known classification methods. We have implemented our approach into the automatic data analysis platform SPIDA developed at BT, which enables non-expert users to run advanced data analysis tasks.

Keywords: Fuzzy decision trees, automatic data analysis, classification models

1 Introduction

Nowadays businesses collect ever increasing amounts of data about internal processes and customer interactions and are faced with the problem of extracting knowledge from that data. For many businesses it becomes unsustainable to employ an increasing number of expensive experts to conduct data analysis manually and they are looking for solutions that can automate and de-skill the data analysis process.

Intelligent data analysis (IDA) is a research area that has its roots in machine learning, statistics, data mining and databases and looks for technologies and strategies to support and to (partially) automate the data analysis and knowledge detection process.

Data analysis has various goals such as segmentation, classification, prediction, dependency analysis etc. Here we concentrate on data analysis in the classification context. A popular form of classification models are decision trees. Classical trees work well in crisp domains, but cannot model vagueness. In order to learn a classification model, which is comprehensible and able to handle vagueness, we have combined fuzzy theory with classical decision trees.

We begin our paper with the introduction of automatic data analysis and the platform SPIDA that was developed in BT's Intelligent Systems Research Centre, for which our fuzzy decision tree module was investigated and implemented. In Section 4 we present our algorithm and examine in detail the core problem of how to compute the information measure in the attribute selection step. Also we discuss how to treat missing values. In addition we consider how to extract a fuzzy rule base from the induced fuzzy tree by which the classification is finally performed and we study heuristics to simplify the rule base. In Section 5 we present our experimental results obtained with an implementation of our algorithm and compare them to those of some popular classifiers. Section 6 concludes the work and points out some further work.

2 Approaches to Automatic Data Analysis

Previous approaches towards automating data analysis or knowledge discovery in databases were based on AI techniques. Analysis methods were broken down into formal blocks and user requirements were also represented in a formal language. Then a search algorithm would identify suitable blocks and arrange them in a way to carry out an analysis process [16].

These approaches had to face the problem of formalising mainly heuristic methods and that it is usually not feasible to formally compute all necessary parameters to execute an analysis

method.

Other authors discussed mainly architectural features of systems that could automate data analysis or data mining and avoided discussing how to automatically select and execute analysis methods [7, 6].

A more recent system uses an ontology based approach and simply describes analysis methods and pre-/post-processing methods as input/output blocks with specific interfaces [2, 1]. The system is built on top of the data analysis package Weka [17]. If the interfaces between two blocks match, they can be concatenated in an analysis process. If a user wants to analyse a data set, all possible analysis processes are created and executed. Once a suitable analysis process has been identified, it can be stored, re-used and shared.

The authors suggest a heuristic ranking of analysis processes in order to execute only the best processes. However, they only use speed as a ranking criterion, which can be easily determined as a feature of an algorithm. More useful features about the quality of the analysis like accuracy are obviously dependent on the analysis process as well as the analysed data and are much more difficult to determine up front. Therefore, the reported results in this area are not very encouraging.

In the Computational Intelligence Group (CI Group) of BT's Intelligent Systems Research Centre (ISR Centre) we have followed a different approach. Especially in industrial settings it is important to empower non-expert users in coping with daily data analysis tasks. 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.

The approach to automating IDA at BT's ISR Centre is based on the following premises:

- User requirements are fuzzy. User's demand *fast, accurate, simple, inexpensive* solutions. All these terms are fuzzy in nature. We have therefore decided to use fuzzy systems for modelling user requirements.

- Most advanced IDA technologies like neural networks, decision trees, neuro-fuzzy system, cluster analysis etc. are all based on heuristics. Based on fuzzy user requirements, it

is basically impossible to formally determine exact values for parameters of those methods. We have therefore decided to use fuzzy systems to determine parameters of analysis methods.

- Properties of analysis methods are fuzzy. Some methods are *fast*, some can produce *accurate* results, others are *interpretable*. Such properties are relevant for their application and are best described by fuzzy sets, since many of them are inherently fuzzy.

- Expert knowledge on how to use which analysis method is vague (fuzzy). 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. We have therefore decided to use fuzzy systems to model such expert knowledge, because this enables us to encode both exact and vague knowledge.

3 SPIDA – A Platform for Automatic IDA

Based on the requirements set above the CI Group at BT's ISR Centre developed SPIDA (Soft computing Platform for Intelligent Data Analysis) [12]. 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, visualisation capabilities and access to different data sources (text files, databases).

The main user group targeted by SPIDA are domain experts. They typically are 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. To achieve this, SPIDA uses fuzzy knowledge bases to match fuzzy user requirements against method features (each analysis method that is available in SPIDA is described by fuzzy and non-fuzzy features) and to automatically determine runtime parameters.

SPIDA is implemented in an open client/server architecture. The server runs IDA processes and the client functions as a graphical user interface (GUI) and can connect both to local and

remote SPIDA servers. A SPIDA server provides a plug-in API that can be used to connect basically any data analysis method or software to SPIDA. Connection is possible, for example, by direct method invocation, or by automatically creating control files that are then executed by a connected software tool in a separate thread. After a new method has been described in the knowledge base, SPIDA is ready to use it.

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.
- 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. Profiles comprise user-specific requirements for IDA processes. They are adapted according to the acceptance or rejection of IDA results.
- 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 standardised 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 [12, 13]

4 Fuzzy Decision Trees

In the past several variants of fuzzy decision trees were introduced by different authors. Boyen and Wenkel [8] presented the automatic induction of binary fuzzy trees based on a new class of discrimination quality measures. Janikow [9] adapted the well-known ID3 algorithm so that it works with fuzzy sets. In this paper, we also adapted the ID3 algorithm to construct fuzzy decision trees and borrowed some basic ideas from [9]. Going beyond Janikow's work we consider how to extract a fuzzy rule base from the resulted fuzzy tree and optimize this rule-based classifier.

The fuzzy decision tree module here has been developed for SPIDA. In this implementation, the configuration for the underlying program is done through the GUI (some of it will be shown in the following), which is running on the client side; and the algorithm for model learning is resident on the server side.

4.1 Tree Induction

In this paper we focus on the induction of a fuzzy decision tree (FDT) 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.

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. We assume the basic algorithm to be known, and only point out some particular features of the fuzzy tree learning as following:

1. **The membership degree of examples**

   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 and its membership degrees to the classes, where different t-norms ($\top$) 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. **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 add a novel 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.

4.2 Information Measures: The Problems And Extensions

A standard method to select a test attribute in classical decision tree induction is to choose the attribute that yields the highest information gain. In this subsection we discuss the problems that occur if we apply this measure directly in fuzzy decision tree induction.

The definition of information gain is based on probability theory. Its value, as we know, can never be negative (a prove can be found, for instance, in [5]). However, depending on the computation, in FDT negative information gain is possible. 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 the 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 $\top_{\min}(a, b)$ and $\top_{\text{Łuka}}(a, b)$.

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 [14], be modified as following:

$$\begin{aligned} & \widehat{\mathrm{Gain}}(\chi^N, A_i) \\ &= \text{frequency of examples with known } A_i \cdot (I(\chi^N) - I(\chi^N|A_i)) \\ &+ \text{frequency of examples with unknown } A_i \cdot 0 \\ &= \alpha \cdot (I(\chi^N) - I(\chi^N|A_i)), \text{ where } \alpha = \frac{Z^{N|A_i \text{ known}}}{Z^N} \end{aligned} \tag{2}$$

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:

$$\mathrm{GainR}(\chi^N, A_i) = \alpha \cdot \frac{I(\chi^N) - I(\chi^N|A_i)}{\mathrm{SplitI}(\chi^N, A_i)} \tag{3}$$

Since the split information $\mathrm{SplitI}(\chi^N, A_i)$ 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 $\mathrm{SplitI}(\chi^N, A_i)$ 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.

4.4 Fuzzy Rule Base

An important goal of this work is to learn a comprehensible classification model, here a fuzzy rule base, which can be generated from the induced fuzzy tree by transforming each path to a leaf 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.

A simpler model or a model with better predictive power cannot be produced by such rewriting of the tree. To achieve this an optimization of the rule base is necessary. We optimize the

rule base by rule pruning, where three heuristic strategies are used, which are adapted from [11]:

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.

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. We do not present here how to use the fuzzy rule base to perform the classification on new data, since it works in basically the same manner as standard fuzzy systems.

5 Experiments

In this section, we report some results obtained from experiments run with the program FDT[2], which was written by the first author of this paper, the well-known decision tree learner C4.5 (Release 8)[3] [14], a neural network training program [4], and NEFCLASS [10], 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 [3]. Table 2 shows general information about these data sets.

All experiments were run with 10-fold 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 1000 epochs.

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

[3]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.

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 2: test data

model		*iris*	*glass*	*thyroid*	*wbc*	*pima*
FDT (1)	$\bar{\varepsilon}$	4.67%	34.29%	**3.33%**	2.79%	31.32%
	n	3	11	5	12	2
FDT (2)	$\bar{\varepsilon}$	5.33%	31.90%	7.62%	2.64%	**18.82%**
	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%

Table 3: 10-fold cross validation

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.

5.1 Precision And Complexity

Table 3 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 3 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 probability 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.

5.2 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.

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

Table 4: Learning from data with missing values

Two columns of Table 4[4] 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.

6 Conclusions

Intelligent data analysis is playing an increasingly important role in businesses and industries. Focusing on the classification problem, we extended classical decision trees by means of fuzzy methods, to achieve comprehensible classifiers with the abilities of modelling vagueness, and implemented our approach into the automatic data analysis tool SPIDA.

Although the learning principle of FDT is the same as that of classical decision trees, it was

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

nevertheless strongly influenced by fuzzy theory. In particular, in FDT the information measure used for the test attribute selection, because of the properties of fuzzy logic, can become negative. We introduced amendments for two measures—information gain and information gain ratio—to ensure a correct ranking of the candidates. To deal with missing values, we also presented further modifications for these measures. Furthermore we suggested to use a threshold for the information measure, so that the complexity of the tree is able to be better controlled.

To optimize the fuzzy rules we extract from a FDT, we transferred three heuristic pruning strategies from NEFCLASS. Since the rules are expressed linguistically, the classifier is easy to interpret. In our experiments we observed that the approach proposed here often generates smaller and at the same time comparably good rule bases. Hence we conclude that we reached our goal of obtaining comprehensible classifiers.

Future work consists in investigating other fuzzy partitioning techniques to enhance the quality of the initial partitioning.

References

[1] A. Bernstein, S. Hill, and F. Provost. Intelligent assistance for the data mining process: An ontology-based approach. CeDER Working Paper IS-02-02, Center for Digital Economy Research,Leonard Stern School of Business, New York University, New York, 2002.

[2] A. Bernstein and F. Provost. An intelligent assistant for the knowledge discovery process. CeDER Working Paper IS-01-01, Center for Digital Economy Research,Leonard Stern School of Business, New York University, New York, 2001.

[3] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[4] C. Borgelt. `mlp` — multilayer perceptron training (computer software). http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html#mlp.

[5] C. Borgelt and R. Kruse. *Graphical Models - Methods for Data Analysis and Mining*. J.Wiley & Sons, Chichester,England, 2002.

[6] J. A. Botia, A. F. Skarmeta, J. R. Velasco, and M. Garijo. A proposal for meta-learning through a mas. In Tom Wagner and Omer Rana, editors, *Infrastructure for Agents, Multi-*

Agent Systems, and Scalable Multi-Agent Systems, number 1887 in LNAI, pages 226–233, Berlin, 2000. Springer-Verlag.

[7] J. A. Botia, J. R. Velasco, M. Garijo, and A. F. G. Skarmeta. A Generic Datamining System. Basic Design and Implementation Guidelines. In Hillol Kargupta and Philip K. Chan, editors, *Workshop in Distributed Datamining at KDD-98*, New York, 1998. AAAI Press.

[8] X.P. Boyen and L. Wenkel. Fuzzy Decision Tree inducion for power system security assessment. *IFAC Symposium on control of power plants and power systems*, 1995.

[9] C.Z. Janikow. Fuzzy Decision Trees: Issues and Methods. In *IEEE Transactions on Systems*, volume 28, pages 1–14, Budapest, 1998. Man and Cybernetics.

[10] D. Nauck and U. Nauck. Nefclass — neuro-fuzzy classification (computer software). http://fuzzy.cs.uni-magdeburg.de/~nauck/software.html.

[11] D. Nauck, U. Nauck, and R. Kruse. NEFCLASS for JAVA — New Learning Algorithmus. In *In Proc. 18th International Conf. of the North American Fuzzy Information Processing Society (NAFIPS99)*, pages 472–476, New York, NY, 1999. IEEE.

[12] D. Nauck, M. Spott, and B. Azvine. SPIDA – a novel data analysis tool. *BT Technology Journal*, 21(4):104–112, October 2003.

[13] D. Nauck, M. Spott, and B. Azvine. Fuzzy methods for automated intelligent data analysis. In *Proc. Int. Conf. on Fuzzy Systems FuzzIEEE'2004*, Budapest, 2004. IEEE.

[14] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[15] X. Wang and C. Borgelt. Information Measures in Fuzzy Decision Trees. In *Proc. Int. Conf. on Fuzzy Systems FuzzIEEE'2004*, Budapest, 2004. IEEE.

[16] R. Wirth, C. Shearer, U. Grimmer, T.P. Reinartz, J. Schloesser, C. Breitner, R. Engels, and G. Lindner. Towards process-oriented tool support for knowledge discovery in databases. In *Principles of Data Mining and Knowledge Discovery. First European Symposium, PKDD '97*, number 1263 in Lecture Notes in Computer Science, pages 243–253, Berlin, 1997. Springer-Verlag.

[17] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann Publishers, San Francisco, CA, 2000. Software available at http://www.cs.waikato.ac.nz/~ml/.