Measuring Interpretability in Rule-based Classification Systems

Detlef D. Nauck
Computational Intelligence Group, Intelligent Systems Lab
BTexact Technologies, Adastral Park
Martlesham Heath, Ipswich IP5 3RE, United Kingdom
Tel: +44.1473.605661, Fax: +44.1473.642459, E-Mail: detlef.nauck@bt.com

ABSTRACT

The "unique selling point" of fuzzy systems is usually the interpretability of its rule base. However, very often only the accuracy of the rule base is measured and used to compare a fuzzy system to other solutions. We have suggested an index to measure the interpretability of fuzzy rule bases for classification problems. However, the index can be used to describe the interpretability of any rule-based system that uses sets to partition variables. We demonstrate the features of the index by using two data sets, one simple benchmark set and a real-world example.

I. INTRODUCTION

In order to transform data into valuable information an intelligent approach to data analysis is required. We do not simply require models for prediction which we would blindly follow and let them determine our business decisions. In order to properly facilitate our decision making processes we also require models for understanding, i.e. interpretable models that provide us with explanations.

In data analysis applications an interpretable model of the data is especially important in areas where

- humans usually make decisions, but where machines can now support the decision making process or even take full responsibility for it,
- prior knowledge is to be used in the data analysis process and the modification of this knowledge by a learning process must be checked,
- solutions must be explained or justified to non-experts.

For generating an interpretable model automatically from data, we must be able to measure its interpretability. This measure should at least be useful for ranking different models such that a data analysis process can pick a model with a higher ranking without user intervention.

The application that we discuss in this paper belongs to the last area. At BTexact we have developed ITEMS, an intelligent data analysis system for estimating and visualizing the travel patterns of a mobile workforce. The application allows users to track journey times and to generate explanations why a particular journey was possibly late, for example. We use the neuro-fuzzy approach NEFCLASS to generate such explanations.

In this paper we discuss some aspects of interpretable fuzzy systems (Section II) and consider the special circumstances of generating models for explanation instead of prediction. Then we suggest an interpretability index (Section III) and discuss its features. In Section IV we show how to use NEFCLASS to build explanatory rules and in Section V we present a real world example before we conclude the paper.

II. INTERPRETABILITY IN FUZZY SYSTEMS

The "unique selling point" of fuzzy systems is usually that they are capable of producing simple rule-based solutions using simple terms. However, in the literature most applied fuzzy systems are evaluated according to their performance or accuracy. Especially, data-driven fuzzy systems, i.e. fuzzy systems generated by learning algorithms focus on accuracy. This is obviously the easiest way for comparing fuzzy systems to other solutions that are based on different approaches.

In order to evaluate fuzzy systems in an area where they are supposed to excel we need a way to assess what we could call their interpretability, simplicity or user friendliness. In this paper we suggest a way of measuring the interpretability of a fuzzy system. We will focus on data-driven fuzzy systems that are generated by, for example, neuro-fuzzy approaches. We assume that the purpose of a fuzzy system generated in a learning process is not only to provide prediction but also to generate explanations about the data set.

If we want a fuzzy system to provide explanatory rules for a given data set then we must consider the following issues.

- What does interpretability mean in the context of fuzzy sets?
- What kind of explanations does the user want and can understand?
- How can we build such rules quickly in interactive applications?
- How do we present the explanations?

We have already discussed several aspects about the interpretability of fuzzy in other publications [5], [8], [9], [10], [11], [12]. This topic is discussed for some time in the literature [1] and more studies about the interpretability of fuzzy systems are being published [2].

In this paper we only consider Mamdani-type fuzzy rules of the form

$$\text{if } x \text{ is } \mu \text{ and } \ldots \text{ and } x_n \text{ is } \mu_n \text{ then } \ldots,$$

i.e. fuzzy rules that use operators like "or" and "not" or lin-

guistic hedges are not discussed here. The consequent of a fuzzy rule can be a linguistic term (regression) or a class label (classification). In this paper we only consider systems classifications, i.e. we do not discuss the interpretability of rule consequences. The following list describes a number of desirable features of an interpretable fuzzy rule base.

- Small number of fuzzy rules.
- Small number of variables used in each rule.
- Small number of fuzzy sets per variable (coarse granularity).
- Unambiguous representation of linguistic terms, i.e. only one fuzzy set for each linguistic term.
- No completely contradictory rules, i.e. there is no pair of fuzzy rules (A, C) and (A', C') with

$$A = A' \wedge C \neq C',$$

where A, A' are antecedents and C, C' are consequents. However, partial contradiction

$$(A \not\subseteq A')A \cap A' \neq \emptyset \wedge C \neq C'$$

is a feature of fuzzy rules and is therefore acceptable.

- No redundancy, i.e. there is no pair of rules (A, C) and (A', C') with

$$A \subseteq A' \wedge C = C'.$$

- Avoiding exceptions, i.e. there should not be a pair of rules (A, C) and (A', C') with

$$A \subseteq A' \wedge C \neq C'.$$

Although exceptions can be a way to reduce the number of rules they can make the rule base harder to read.

- Fuzzy partitions should be "meaningful", i.e. fuzzy sets should be normal and convex and must keep their relative positions during learning.

III. An Interpretability Index

When fuzzy systems are generated in a learning process, for example in a data analysis scenario, the usual target is to build a reasonably accurate predictor with a reasonably interpretable rule base. Obviously, there is a trade-off between accuracy and interpretability and depending on the application the solution will concentrate more on prediction than interpretability, or vice versa.

In order to obtain a model that can be used for prediction we have to avoid over-generalization. This means, we usually split up the available data in training and validation sets. In order to obtain a useful estimate of the error on unseen data this procedure is usually repeated several times (n-fold cross-validation).

We can also build a model solely for the purpose of explaining patterns in a data set and then we have a different target. The model will not (and must not) be used for prediction. The idea of the model is to provide a summary of the currently available data in a meaningful way. In this sense an explanatory model is like a descriptive statistic of a sample, where we are only interested in describing the sample and not the underlying population.

That means to build an explanatory model we will use all available data and do not worry about over-generalization. Actually, we try to get the best possible fit to the data because that means we obtain the most accurate description of the data. We will also restrict the degrees of freedom for our model in order to obtain a small, simple and interpretable model. Obviously this leads to the same dilemma as for predictive models: the trade-off between accuracy and interpretability.

The demand for a predictive model usually does not impose strict time constraints on the model building process. The only real requirement is that the model is available before the first prediction is due. Explanations, however, are more likely to be demanded in a real time process. A typical scenario is that a user explores some data and demands an explanation for some findings. It is then not acceptable to wait a long time for a model. The model building process must be completed within seconds or minutes at most.

This time constraint forbids an extensive search for the smallest, most interpretable model with an acceptable accuracy. That means in a real world applications there is no typically time for building a model, checking its interpretability and then restart the model building with different parameters until we have reached the best result. However, it may be possible to do this in parallel, if the computing resources are available.

Therefore, we are interested in learning approaches that either try to build small models from the beginning or are capable of pruning a large model as part of the learning process.

Rule induction methods based on decision trees try to build small models by selecting the most promising variables first in the hope of not requiring all variables for the final model [4], [14]. Another option is to use a hyperbox-oriented fuzzy rule learning method [11] as it is implemented in the neuro-fuzzy system NEFCLASS. This approach first detects all rules that are induced by a data set, selects the best rules based on a rule performance measure, trains the fuzzy sets, and then prunes the rule base. For ITEMS we have implemented explanation facilities based on (crisp) decisions trees and neuro-fuzzy rules. Users can select which version of rules they prefer.

If enough computing resources are available, it may be possible to generate several different explanatory models in parallel. The system can then try to measure the interpretability based on approaches like the minimum description length principle [6] that can also be applied to neuro-fuzzy learning [5]. Another possibility to compare different rule based models is to consider the number of relevant parameters. In the application that we discuss in this paper we are interested in classification rules. We can measure the complexity comp of a classifier by

$$\text{comp} = m \Big/ \sum_{i=1}^{r} n_i \ ,$$

where m is the number of classes, r is the number of rules and n_i is the number of variables used in the ith rule. This

measure is 1, if the classifier contains only one rule per class using one variable each and it approaches 0 the more rules and variables are used. We assume that at least one rule per class is required, i.e. systems with a default rule must be suitably recoded.

When we want to compare fuzzy rule bases we also want to measure the quality of the fuzzy partitions. Usually, a fuzzy partitioning is considered to be "good", if it provides complete coverage (i.e. membership degrees add up to 1 for each element of the domain) and has only a small number of fuzzy sets. If we assume that the domain X_i ($i \in \{1,\ldots,n\}$) of the ith variable is partitioned by p_i fuzzy sets $\mu_i^{(1)},\ldots,\mu_i^{(p_i)}$ then we can measure the degree of coverage provided by the fuzzy partition over X_i by the coverage index

$$\begin{aligned} \mathrm{cov}_i &= \frac{\int_{X_i} \hat{h}_i(x)dx}{N_i}, \text{where} \qquad (1)\\ \hat{h}_i(x) &= \begin{cases} h_i(x) & if \;\; 0 \le h(x) \le 1 \\ \dfrac{p_i - h_i(x)}{p_i - 1} & \text{otherwise} \end{cases}, \text{where} \\ h_i(x) &= \sum_{k=1}^{p_i} \mu_i^{(k)}(x) \end{aligned}$$

with $N_i = \int_{X_i} dx$ for continuous domains. For discrete finite domains we have $N_i = |X|$ and replace the integral in (1) by a sum.

We can see that $\mathrm{cov}_i \in [0,1]$, where $\mathrm{cov}_i = 0$ means that the variable is either not partitioned or that we have a crisp partition such that all $\mu^{(i)} = X_i$. Both extreme cases mean that the variable would be considered as 'don't care' and would not appear in any rule. Complete coverage is indicated by $\mathrm{cov}_i = 1$. Note that a partition that covers only 10% of all values gets approximately the same score as a partition where 90% of the whole domain is completely covered by all sets. This feature may be disputable and has to be considered when applying the index.

In order to penalize partitions with a high granularity we can use the partition index

$$\mathrm{part}_i = \frac{1}{p_i - 1}$$

assuming that $p_i \ge 2$, because otherwise we would not consider that variable. We use $\overline{\mathrm{cov}}$ to denote the average normalized coverage and $\overline{\mathrm{part}}$ to denote the average normalized partition index for all variables actually used in the classifier.

Finally, we can use the interpretability index

$$I = \mathrm{comp} \cdot \overline{\mathrm{cov}} \cdot \overline{\mathrm{part}}$$

for measuring the interpretability of a fuzzy classifier. Obviously, we can apply I to any rule-based classifier that uses sets to partition variables, by simply representing crisp sets by corresponding fuzzy sets.

A classifier would only score $I = 1$ if it contains one rule per class, using only one variable per rule and providing complete coverage for each variable with two (fuzzy) sets and $m \le 2n$ holds, where n is the number of variables. The value of I can give us an idea if we want to compare two rule-based classifiers for the same problem. It is less useful to compare the interpretability of classifiers for different domains.

We also have to point out that for an explanatory model local interpretability can be more important than global interpretability. If, for example, an explanatory model is used for obtaining explanations only for individual patterns and not for obtaining an understanding of the data set as a whole, then the number of rules that fire for any given pattern are of interest and not the overall number of rules in the rule base. In this case the complexity index comp could be replaced by an average value based on the rules that fire for a selected pattern.

IV. Creating Interpretable Rule Bases in NEFCLASS

The NEFCLASS learning structure and parameter learning algorithms are designed in a way that they can operate fully automatically if this is required. Usually, if interpretable solutions are required from a neuro-fuzzy system, then user interaction is a desired feature, because only the user can decide what "interpretable" means in a certain application scenario. If, however, a fuzzy rule base shall be created within an application, then we cannot assume that the user is willing or capable of supervising a neuro-fuzzy learning process. Therefore it must be possible to create a rule base completely automatically while trying to obtain a high degree of interpretability. NEFCLASS uses the following strategies in order to achieve this goal.

- **Automatic best per class rule learning**: this feature creates a rule base that contains so many rules that all patterns of the training set are covered by at least one rule. In the first stage NEFCLASS creates a new rule each time it encounters a pattern that is not yet covered by a rule, i.e. if the pattern does not have a degree of membership of at least 0.5 with any rule antecedent currently stored in the rule base. During the second stage NEFCLASS selects for each class the best rules for the final rule base. It does that by selecting a rule for each class in a round-robin fashion until all patterns are covered by at least one rule. This algorithm also guarantees a similar number of rules for each class.
- **Automatic fuzzy set tuning**: the fuzzy set learning algorithm uses a heuristic to modify the membership function in order to reduce the sum of squared errors (SSE) and the number of misclassifications. A small overall SSE usually indicates that the classifications are nearly crisp and a small number of misclassifications is an obvious target. In order to obtain meaningful fuzzy sets, the learning algorithm is constrained. For example, fuzzy sets must not change their relative position to each other and must always overlap.
- **Automatic exhaustive pruning**: NEFCLASS includes four pruning strategies that try to delete variables, rules, terms and fuzzy sets from the rule base. In order to

obtain a rule base that is a small as possible all those four strategies are applied one after the other in an exhaustive way to all possible parameters. For the sake of speed the fuzzy sets are not retrained after each pruning step, but only once after pruning.

In order to start the learning process NEFCLASS requires initial fuzzy partitions for all variables. This part is not yet fully automated, because for numeric variables a number of fuzzy sets must be specified. For the discussed scenario this should be done by the application designer. We are currently working on automating this process as well. For symbolic variables, NEFCLASS can determine the initial fuzzy partitions automatically during rule learning [7].

When we execute NEFCLASS in an application environment with no user intervention, then we must try to balance the required computation time with the quality of the rule base. While rule learning and pruning are usually fast, fuzzy set learning can take a lot of time, because it requires many iterations through the training data set. Fuzzy set learning is influenced by some parameters (learning rate, batch/online learning, look ahead mode for trying to escape local minima) and by the already mentioned constraints for guaranteeing meaningful fuzzy sets.

For running fuzzy set learning automatically we can select a small learning rate (e.g. 0.1) together with batch learning to avoid oscillation and select a small look ahead value (10 epochs) which continues training beyond a local minimum in the hope to escape it again. The number of epochs are usually chosen in such a way that a minimum number of learning steps are computed (e.g. 100) and that learning is stopped after, for example, 30 seconds. This time has to be set according to the application scenario. If users of the application usually tend to inspecting other features of the requested data first before checking the fuzzy rules, learning may continue for longer. If the user mainly waits for the rules, learning may have to be shorter.

We also must balance the constraints we impose on the learning algorithm. We could enforce strict constraints like that the membership degrees for each element must add up to 1.0 [11], [13]. However, strict constraints tend to prevent the system from reaching a acceptable classification performance and usually require inspection of the learning outcome and repeated trials, for example, with different numbers of fuzzy sets. In an automated scenario this is not possible and we would use only the above-mentioned less strict constraints.

In order to illustrate how interpretable fuzzy rule bases generated by NEFCLASS are we may consider the Iris data set as a very simple, but frequently used benchmark. If we force NEFCLASS to create fuzzy partitions with complete coverage (membership degrees add up to 1 for each element of the domain) then NEFCLASS can create the following rule base for the Iris data set after pruning:
if petal width is small then Iris setosa,
if petal width is medium then Iris versicolour,
if petal width is large then Iris virginica.

On our suggested interpretability index this rule base scores

$$\begin{aligned} \text{comp} &= \frac{3}{1+1+1} = \\ \overline{\text{cov}} &= 1 \text{ (we enforced complete coverage)} \\ \overline{\text{part}} &= \frac{1}{3-1} = 0.5 \text{ (only one variable is used)} \\ I &= 0.5 \end{aligned}$$

The rule base was generated on 50% of the data (2 errors), while the remaining 50% were used for testing (4 errors). On the complete set (150 records) we obtain a performance of 96%.

If we generate a decision tree on the complete data set using information gain ratio and transform this tree into a rule base, then after pruning the tree we obtain 6 rules using one or two variables. The variable petal length is partitioned into four intervals and the variable petal width into three intervals. This rule base has a performance of 98.7% (2 errors) on the whole data set and its interpretability scores as follows:

$$\begin{aligned} \text{comp} &= \frac{3}{1+2+2+2+2+2} = 0.27 \\ \overline{\text{cov}} &= 1 \text{ (crisp subsets)} \\ \overline{\text{part}} &= \frac{1/(3-1)+1/(4-1)}{2} = 0.42 \\ I &= 0.11 \end{aligned}$$

This very simple example illustrates that fuzzy rule bases can indeed be measurably simpler than rule bases of other approaches. On the other hand we have to expect a slight reduction in accuracy. The suggested interpretability index can illustrate this feature of fuzzy rule bases and it is especially useful for selecting an appropriate rule base when user interaction is not possible. If users can be involved in the rule generation and selection process then they will usually select a result based on their personal preferences. However, even in such a scenario an interpretability index can be useful in order to rank different results.

In the following section we will explore the features of the interpretability index by using a real-world example.

V. Explaining Travel Patterns of a Mobile Workforce

Any organization with a large mobile workforce needs to ensure efficient utilization of its resources as they move between tasks distributed over a large geographical area. BT employs around 20000 engineers in the UK who provide services for business and resident customers such as network maintenance, line provision and fault repairs. In order to manage its resources efficiently and effectively, BT uses a sophisticated dynamic scheduling system to build proposed sequences of work for field engineers.

A typical schedule for a field engineer contains a sequence of time windows for travel and task. To generate accurate schedules the system must have accurate estimates for the time required to travel between tasks and estimates for task

duration. At BTexact Technologies – BT's advanced communications technology business – we have implemented a system that improves the accuracy of travel time estimates by 30% compared to the previous system. The system [3] contains an estimation module, a visual data mining module and an explanation facility.

Note that the travel time is calculated as the difference between the time a task is issued and the arrival time on site. Specifically, travel time includes the time required to leave the current site, walk to the car-park, start the car, drive to the destination site, park the car, and gaining access to the premises of the next customer. However, all these individual activities are not logged, only the start time of the journey and the start time of the actual work are available. It is obvious that we have to deal with huge differences between urban and rural areas, for example, just in finding a space to park.

The Intelligent Travel Estimation and Management System (ITEMS) is a web-enabled, Java-based software system that predicts, manages, visualizes and explains travel patterns of a mobile workforce. ITEMS is a tool for service industries like telecommunications, gas, water, electricity etc. that have to schedule jobs for large mobile workforces. Successful scheduling requires suitable estimates of inter-job times that are mainly determined by travel time. However, it is not sufficient to simply use routing software, because that cannot estimate the time that is required to find a parking space, to gain access to the site etc. It is also impossible for technicians to log detailed information about the routes they have taken, but they only log their actual travel (inter-job) time. Thus, it is not possible to compare actual travel data with recommendations from routing software.

ITEMS has a learning component that constantly builds new models for travel time prediction. It compares the new model with the performance of the model currently used by the scheduler and recommends updating the scheduler if the new model performs significantly better. The learning component makes sure, that the scheduler will gradually adapt to changes in travel behaviour. So if, for example, technicians are frequently late on specific journeys, the scheduler will obtain new travel time estimates for those journeys after a sufficient amount of data has been gathered. It can then compute better schedules and try to avoid those critical journeys. While updating the travel estimates may take time, managers can still quickly react to critical situations by making use of the visualization and explanation facility of ITEMS.

In addition to reliable estimates workforce managers also regularly need to analyze the travel behaviour of their workforce in order to determine if improvements are required. ITEMS provides a color-coded geographical visualization of travel patterns. Managers can easily identify areas where travel is slow and can assess the performance of technicians on a weekly, daily and individual basis.

ITEMS contains an explanation facility based on decision trees and neuro-fuzzy systems that display rule-based information about individual journeys. The rules derived from travel data explain, for example, why a certain journey may have been late. The information of those rules can be used by managers to improve the overall system behaviour. Let us, for example, assume that the automatically generated rules reveal that travel between two specific areas takes usually longer than predicted at a certain time of day. In this case, the scheduler can be advised and try avoid scheduling journeys between those two areas at that time of day.

The purpose of the explanatory rules is to provide resource managers with a tool to investigate workforce travel patterns. The rules provide a summary of the actual data and highlight influential variables. In order to be useful the rules must be simple and sparse. It must also be possible to create the rule base on the fly in a very short time. The user would select a specific set of journeys for which he requires explanations. The system must then create the rules completely automatically without user interaction.

For the explanation module we can use a decision tree learner or the NEFCLASS algorithms to generate rules. Figure 1 displays a screen shot of typical situation while using ITEMS. A user analyzes the travel pattern of some technician and has clicked an arrow in the displayed map. Additional windows display detailed information about the corresponding job. The top right window displays two fuzzy rules that match the travel information of the selected job. The user can see the degree of fulfilment of a rule and decide, if a particular rule is useful to explain the selected travel pattern, i.e. why the technician was late, early or or on time.

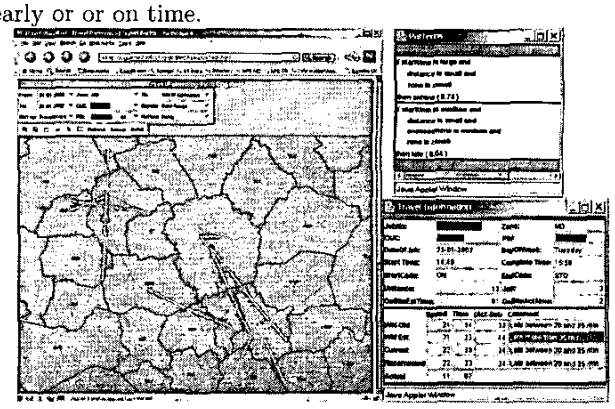

Fig. 1. Screen shot of ITEMS displaying two fuzzy rules

In the following we present a small example of how NEFCLASS can generate rules for explaining travel patterns. The data contains both numeric and symbolic data and we are using the algorithm described in [9]. For confidentiality reasons we can only reveal parts of the result.

As an example we take a closer look at one model for one organizational unit comprising 13 technicians, using three weeks of data. This is a typical set of data analyzed by a unit manager. The data contains 10 input variables, where five are symbolic. The data was classified into three classes: early, on time and late. After rule learning, fuzzy set tuning and pruning, NEFCLASS presented a rule base of seven rules using four variables (Technician, Time (hour), Start

Location, Destination).

The accuracy of the fuzzy rule base was 76.6% with the following class accuracies: early = 30%, late = 32%, on time = 97% As a comparison we created a decision tree with 66 rules with more than six variables on average that was 70% accurate. The fuzzy rule base obtained a score of $I = 0.0066$ (comp = 0.1875, $\overline{\text{cov}} = 0.15$, $\overline{\text{part}} = 0.2375$) on our interpretability measure introduced in Section III, while the decision tree has $I = 0.0025$ (comp = 0.0076, $\overline{\text{part}} = 0.33$, $\overline{\text{cov}} = 1$).

Let us concentrate on the fuzzy rules that describe late journeys for this particular set of jobs. Group_n, Start_n and and End_n are list fuzzy sets for the symbolic variables ID (technician), Start (start location) and End (destination), where n is an index.

- If ID is Group_1 and Start Hour is small and Start is Start_1 and End is End_1 then Late
- If ID is Group_4 and Start is Start_2 and End is End_3 then Late
- If ID is Group_5 and Start is Start_3 and End is End_4 then Late

Further analysis of the classification of individual jobs shows that 85% of the misclassified *late* patterns have almost equal membership with *late* and *on time*. In addition all *late* patterns have actually non-zero membership with the class *late*.

When the user clicks on the graphical representation of a journey in the graphical user interface, he will see all rules that fire for that particular record. If the user is particularly interested in late travel, at least one of the above presented rules will be displayed. Even if the pattern is misclassified, the rules can still provide a useful explanation.

Note, that the rules are not meant for prediction, but for explanation only. That is why we use all the data to generate the rules and do not use validation. The rules represent a rule-based summary of the data. For example, if there is a rule that correctly classifies a lot of late journeys, the manager can investigate, why this particular pattern is present in the data. Such a scenario can point to problems in the prediction module or to specific situations like ongoing road works that are only indirectly reflected in the data. On the other hand, if there is a late journey that cannot be classified correctly, this means that it cannot be explained by the data and may be an outlier (exception) so that no action is required.

Most of the rules shown above contain only symbolic variables and therefore list fuzzy sets. They are not as easily interpretable as, for example, the fuzzy set *small* for the variable Start Hour in the first rule. However, close inspection of the fuzzy sets reveals, that for example technicians who were frequently late all have high degrees of membership with the fuzzy sets Group_1, Group_4 and Group_5. This can help the manager in identifying, for example, individuals who may be require additional training or information about better routes, because they are, for example, new on the job.

VI. Conclusions

We have suggested an interpretability index to measure the simplicity and readability of a (fuzzy) rule based solution. This index can be used in data analysis process to select an appropriate solution when user interaction is not possible or suitable.

Intelligent data analysis plays a crucial role in modern businesses. They do not only require predictions based on data but also require a deep understanding of the data that is collected from internal or external sources. Rule based models that provide explanations can be a valuable tool in this area. We have shown how we have used the neuro-fuzzy approach NEFCLASS in the an application scenario where explanatory rules are required. The learning algorithms of NEFCLASS are capable of generating explanations about a data set selected by a user in a reasonably short time.

References

[1] Hugues Bersini, Gianluca Bontempi, and Mauro Birattari. Is readability compatible with accuracy? From neuro-fuzzy to lazy learning. In *Fuzzy-Neuro Systems '98 – Computational Intelligence. Proc. 5th Int. Workshop Fuzzy-Neuro-Systems '98 (FNS'98) in Munich, Germany*, volume 7 of *Proceedings in Artifical Intelligence*, pages 10–25, Sankt Augustin, 1998. infix.

[2] J. Casillas, O. Cordon, F. Herrera, and L. Magdalena, editors. *Trade-off between Accuracy and Interpretability in Fuzzy Rule-Based Modelling*. Studies in Fuzziness and Soft Computing. Physica-Verlag, Heidelberg, 2002.

[3] Colin Ho and Ben Azvine. Mining travel data with a visualiser. In *Proc. International Workshop on Visual Data Mining at the 2nd European Conference on Machine Learning (ECML'01) and 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)*, Freiburg, 2001.

[4] Cezary Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Trans. Systems, Man & Cybernetics. Part B: Cybernetics*, 28(1):1–14, 1998.

[5] Aljoscha Klose, Andreas Nürnberger, and Detlef Nauck. Some approaches to improve the interpretability of neuro-fuzzy classifiers. In *Proc. Sixth European Congress on Intelligent Techniques and Soft Computing (EUFIT98)*, pages 629–633, Aachen, 1998.

[6] I. Kononenko. On biases in estimating multi-valued attributes. In *Proc. 1st International Conference on Knowledge Discovery and Data Mining*, pages 1034–1040, Montreal, 1995.

[7] Detlef Nauck. Using symbolic data in neuro-fuzzy classification. In *Proc. 18th International Conf. of the North American Fuzzy Information Processing Society (NAFIPS99)*, pages 536–540, New York, NY, 1999. IEEE.

[8] Detlef Nauck. Adaptive rule weights in neuro-fuzzy systems. *Neural Computing & Applications*, 9(1):60–70, 2000.

[9] Detlef Nauck. Fuzzy data analysis with nefclass. In *Proc. Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, pages 1413–1418, Piscataway, July 2001. IEEE.

[10] Detlef Nauck. Neuro-fuzzy systems for explaining data sets. In *Proc. NAFIPS-FLINT 2002 International Conference*, pages 195–200, New Orleans, June 2002.

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

[12] Detlef Nauck and Rudolf Kruse. How the learning of rule weights affects the interpretability of fuzzy systems. In *Proc. IEEE Int. Conf. on Fuzzy Systems 1998*, pages 1235–1240, Anchorage, May 1998.

[13] Detlef Nauck and Rudolf Kruse. NEFCLASS-J – a java-based soft computing tool. In Behnam Azvine, Nader Azarmi, and Detlef Nauck, editors, *Intelligent Systems and Soft Computing: Prospects, Tools and Applications*, number 1804 in Lecture Notes in Artificial Intelligence, pages 143–164. Springer-Verlag, Berlin, 2000.

[14] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.