RULES-F: A fuzzy inductive learning algorithm

D T Pham, S Bigot and S S Dimov

Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 2006 220: 1433
DOI: 10.1243/0954406C20004

RULES-F: a fuzzy inductive learning algorithm

D T Pham*, **S Bigot,** and **S S Dimov**
Intelligent Systems Research Laboratory, Manufacturing Engineering Centre, Cardiff University, Cardiff, UK

The manuscript was received on 22 November 2004 and was accepted after revision for publication on 29 March 2006.

DOI: 10.1243/0954406C20004

Abstract: Current inductive learning algorithms have difficulties handling attributes with numerical values. This paper presents RULES-F, a new fuzzy inductive learning algorithm in the RULES family, which integrates the capabilities and performance of a good inductive learning algorithm for classification applications with the ability to create accurate and compact fuzzy models for the generation of numerical outputs. The performance of RULES-F in two simulated control applications involving numerical output parameters is demonstrated and compared with that of the well-known fuzzy rule induction algorithm by Wang and Mendel.

Keywords: fuzzy systems, rule induction, inductive learning, numerical value prediction, control systems

1 INTRODUCTION

Many engineering applications require the creation of models for the prediction of variables with numerical values. Techniques have been developed to build approximate models for such a purpose. One of the most popular methods is the generation of models in the form of neural networks. This requires effort in selecting the structure of the neuron layers and produces 'black box' type models, which are difficult to interpret. A number of studies have been conducted on the adaptation of typical inductive learning algorithms for the handling of numerical outputs in order to provide more transparent models. Many researchers have attempted to use decision-tree induction and have handled numerical output problems by simply dividing the output range into small intervals to be regarded as nominal class values. However, these attempts often fail [**1**] because they yield extremely large or inaccurate models. Successful attempts have tended to employ regression-tree algorithms, the most famous of which perhaps is CART [**2**].

A regression-tree model is generated in a similar way to a typical classification-tree model, and once built, such a model is also similarly interpreted. In CART, for a numerical attribute A^i, each internal node of the tree is a test of the type $[A^i > t^i]$, where t^i is a test value for A^i. The main differences appear in the interpretation of a leaf and in the splitting criterion employed. At a particular leaf, the predicted output value is the mean of the output values of the examples in the training set T reaching the leaf. The splitting process is performed in order to minimize the mean-squared error at each leaf, defined as the average of the squared differences between the actual and the predicted output values of the examples in T reaching the leaf.

On the basis of the CART approach, many other algorithms have been developed. For instance, Morimoto [**3**] presented a more accurate algorithm for the construction of region-splitting regression trees, similar to CART but including a wider range of possible tests for the internal nodes. With another algorithm, Karalic and Cestnik [**4**] introduced the notion of a Bayesian approach to estimate the class distribution in tree-structured regression, which resulted in the modification of the splitting and pruning criteria and permitted the creation of trees with smaller classification errors.

Nevertheless, a problem with these types of regression trees is that they all employ constant values at their leaves. This restricts the use of the regression-tree model when handling prediction problems because the number of possible predicted values will be limited by the number of leaves in

*Corresponding author: Intelligent Systems Research Laboratory, Manufacturing Engineering Centre, Cardiff University, Queen's Building, Newport Road, Cardiff CF24 3AA, UK. email: phamdt@cardiff.ac.uk

the tree. Other attempts were based on covering algorithms, for example, CN4 and KEX [**5**]. However, as in the original regression trees, the value predicted by each rule (equivalent to a leaf) is a single value.

To resolve this problem, a new type of regression method was developed. The method is based on the typical decision-tree structure but it uses local linear-regression functions instead of single values at the leaves [**6**]. A similar idea is adopted in the M5 algorithm [**1**], where at each leaf, the output prediction depends on a linear function with the attribute values as parameters. However, with the use of linear-regression functions, inductive learning algorithms lose the characteristic of being close to human reasoning,which gave them their popularity.

A particular type of model, based upon fuzzy logic, has been adopted increasingly frequently for the development of expert systems and intelligent controllers, because of its similarity to some aspects of human reasoning. Fuzzy logic models combine comprehensibility with the ability to deal with numerical outputs and do not require sophisticated mathematics. A key feature of fuzzy logic models is that they can handle vagueness and uncertainty. 'Fuzzy logic allows one to deal with fuzzy and high-level notions (e.g. being small) while processing low-level information (e.g. size is 3.4 in)' [**7**]. Fuzzy logic becomes useful when processing systems that are too complex for conventional methods or when the available information is qualitative, inexact, or uncertain [**8**]. Such systems are generally designed to be operated by human experts, but because of their complexity and the difficulty, cost, and inconvenience of accessing expertise, many techniques have been proposed to automate them.

The first part of this paper reviews the existing algorithms for automatic fuzzy rule generation and relevant concepts of fuzzy logic used in such algorithms. A technique for covering algorithms allowing the creation of compact fuzzy rule sets is introduced, which avoids the drawbacks of these algorithms. This technique has been applied to RULES-5 [**9**], an algorithm previously developed by the authors, which yields a new fuzzy inductive learning algorithm called RULES-F. The paper describes RULES-F and experiments to compare it against the well-known fuzzy induction algorithm by Wang and Mendel [**10**]. For convenient reference, an outline of RULES-5 is provided in Appendix 2.

2 EXISTING ALGORITHMS FOR AUTOMATIC FUZZY RULE GENERATION

Typically, fuzzy-rule generation is based on a multidimensional matrix representation called a decision table (one dimension per attribute). Each cell in the matrix is called a fuzzy subspace and represents a possible rule when linked with a particular output. However, this method is impractical when the number of attributes increases. Thus, a number of studies have been performed to automate this process. The strategy employed is to replace the expert knowledge by a list of training examples, *T*, representing the behaviour of the domain studied.

A well-known method is that proposed by Wang and Mendel [**10**]. This technique first needs predefined fuzzy membership functions that divide the attribute space into fuzzy regions. On the basis of these membership functions, a fuzzy rule is generated for each example. Each rule is stored in a decision table, and in case of conflict, a degree of truth for each rule is assessed to select the best rule, and therefore, the fuzzy membership function to be stored in the decision table. However, as mentioned by the authors, there is a problem of 'growing memory': when more training examples become available, more rules are generated and the selection of the best rules becomes difficult. To help this selection, Delgado and Gonzalez [**11**] used a method based on the definition of frequencies in each fuzzy domain, which allows one to identify whether any possible rule is a 'true rule'.

Another fuzzy-rule generation method was proposed by Nozaki and Ishibuchi [**8**]. As in Wang and Mendel's algorithm, the membership functions are first defined and then a decision table is formed. The main idea of the method is that using a particular heuristic based on the examples in *T*, it computes single real numbers (instead of membership functions), which are to be stored in each cell of the decision table. Only then are the real numbers transformed into fuzzy output in order to obtain a complete set of fuzzy rules. The problem is that the algorithm will consider as many rules as exist combinations of attributes/membership values, so that the problem of growing memory, mentioned by Wang and Mendel [10], is also present.

These two methods create very large fuzzy rule sets that need to be post-processed in order to obtain more compact sets of rules. Furthermore, for both methods, the membership functions need to be predefined and this could be a difficult task. In fact, the problem of 'designing the membership function may be just as complex as designing fuzzy rules' [**7**]. Sebag and Schoenauer proposed an inductive learning algorithm that enables the automatic creation of membership functions, but the process is still dependent on parameters input by the user, which requires a knowledge of the domain studied.

A handful of other methods have been proposed in order to automate the fuzzification process. However, as Hong and Lee noted [**12**], a common problem with many of these methods is that the membership functions created are 'fixed and equally partitioned'. In

response to this, Hong and Lee proposed a successful method for the automatic membership function design. The output is fuzzified using a clustering procedure that regards examples in T with close output values as belonging to the same fuzzy set and which then assigns appropriate membership functions to represent each fuzzy set. For the fuzzification, each attribute value is assigned an initial membership function, in the form of a triangle of base equals to a small interval predefined by the user. The decision table is then built using the examples in T with their initial membership functions. Finally, merging between the membership functions is carried out to simplify the decision table, new larger membership functions are formed, and the final rule set can then be extracted from the simplified decision table. This technique has subsequently been improved [**13**] but it still can be very computationally expensive when the number of attributes increases. Hong and Chen [**14**] proposed a method to reduce the cost by simplifying the initial membership functions before building the decision table. This proved to be more efficient and yielded higher accuracies, but the process is still computationally costly. However, the result of this method is a set of fuzzy rules where the membership functions have been automatically created and the universes of discourse are not equally partitioned. Nevertheless, the user still needs to input appropriate initial membership functions, which requires a degree of domain knowledge, and this can be difficult when there is a large number of attributes.

Another type of algorithm has been suggested on the basis of machine-learning techniques. For example, Shann and Fu [**15**] created an algorithm using a neural-network structure, which allows the use of the error back-propagation learning algorithm for fuzzy-rule generation. However, one of the drawbacks of the algorithm presented by Shann and Fu is that the membership functions need to be predefined. Many other algorithms using neural networks exist and most of them use a method similar to Shann and Fu's algorithm.

A solution to the problem of automating membership function design is to adapt inductive learning techniques that have solved the similar task of numerical attributes discretization. A number of fuzzy inductive learning algorithms exist, for instance, the FILM algorithm [**16**], which first creates a crisp decision tree using ID3 [**17**] and then applies fuzzification operations to modify it. In this algorithm, the membership functions are created automatically. Fuzzy logic concepts are used to deal with noise, uncertainty, and imprecision in the attribute values. However, like many more recent fuzzy inductive learning methods [**18–21**], this technique has been developed only for classification problems.

In fact, no suitable fuzzy inductive learning algorithm has been found so far, which has been developed for numerical output prediction and enables the automatic creation of membership functions.

3 FUZZY LOGIC CONCEPTS

In fuzzy rule induction, fuzzy rules are created from a set of training examples (T). Each example E is described by its output value V_E^{out} and by a vector of m attributes ($A^1, \ldots, A^i, \ldots, A^m$). Each attribute value V_E^i is either nominal or numerical. In the case of a numerical attribute, $[V_{\min}^i \leq V_E^i \leq V_{\max}^i]$, where $V_{\min}^i$ is the minimum known value for the ith attribute and $V_{\max}^i$ its maximum known value. An example E is therefore formally defined as follows

$$E = (A^1 = V_E^1, \ldots, A^i = V_E^i, \ldots, A^m = V_E^m, \quad \text{Output} = V_E^{out}) \tag{1}$$

A fuzzy rule R is composed of a number of possible fuzzy conditions on each of the m input attributes and an output fuzzy set (F_R^{out}). It can be represented as follows: $\text{Cond}_R^1 \wedge \cdots \wedge \text{Cond}_R^i \wedge \cdots \wedge \text{Cond}_R^m \Rightarrow F_R^{out}$. For the ith attribute A^i, a fuzzy condition has the form $[A^i \text{ is } F_R^i]$. A number of different shapes can be used for the fuzzy sets F_R^i but, to facilitate computation, the triangular form is adopted in RULES-F because such a form can be defined as Tr(a,b,c), where ac being the base of the triangle and b the location of its apex (Fig. 1). It should be noted that this fuzzy representation could also be employed for nominal values. This simply requires the kth possible nominal value of the ith nominal attribute A^i (Vd_k^i) to be coded as a numerical value (Vcod_k^i) constituting a singleton fuzzy set in order to obtain a condition of the form $[A^i \text{ is } \text{Vcod}_k^i]$.

The use of a fuzzy rule for output prediction is similar to the application of a rule in inductive learning to classify an example. The main difference is that because of the notion of fuzziness, a particular example E will have a particular 'degree of match' ($\mu\_\text{rule}_R(E)$) with each rule R. This can be used in order to evaluate how likely a rule is to predict accurately the output value of an example. The 'degree of

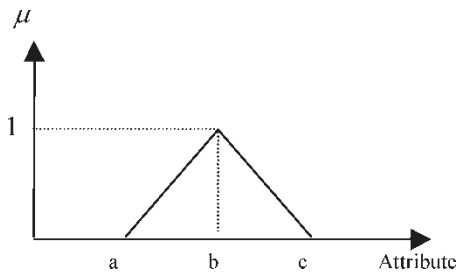


Fig. 1 Triangular fuzzy set

match' is obtained by first assessing the membership degree $\mu_{F_R^i}(V_E^i)$ of each attribute value (V_E^i) of the example with regard to the corresponding condition fuzzy set (F_R^i) in a rule R. Using these membership degrees, the 'degree of match' of an example to each rule can be evaluated. The method adopted for RULES-F uses the product of all membership degrees, namely

$$\mu\_\text{rule}_R(E) = \prod_{i=1}^{m} (\mu_{F_R^i}(V_E^i)) \tag{2}$$

Finally, in inductive learning, when a complete model (a set of rules) has been created from the set of training examples (T), it can then be employed to determine the class of a new example. Different methods could be used to select a specific firing (or covering) rule. In the case of a fuzzy rule set, the idea is similar but the method employed should take into consideration the fuzzy concepts. Moreover, depending on the domain studied, the output can take three different forms: nominal, fuzzy, or numerical. Thus, a fuzzy algorithm will use different methods to select the proper output.

Nominal output. The fuzzy rule set can be applied similarly to a rule set created by a classical inductive learning algorithm. A difference is that in addition to parameters such as the numbers of examples covered, classified, and misclassified by a rule R, the degree of match $(\mu\_\text{rule}_R(E))$ between each example E and R could also be used to evaluate R.

Fuzzy output. In order to obtain a single fuzzy output, a method could be to select the output membership function of the best covering rule. A better method fully taking into consideration the notion of fuzziness is to combine the output membership functions of all covering rules to obtain a new single output fuzzy set.

Numerical output (defuzzification). Most applications require the fuzzy output obtained to be converted into a single numerical value. This process is called defuzzification. A number of different defuzzification methods exist and the reason for choosing one or another method is not always clear. However, Rondeau *et al.* [**22**] demonstrated that the weighted-average method was complementary to triangular fuzzification. Therefore, this defuzzification method was adopted for RULES-F. The weighted-average method uses the following formula to compute the defuzzified output

$$\text{output} = \frac{\sum_{R=1}^{r} [\mu\_\text{rule}_R(E) \cdot C_R^{\text{out}}]}{\sum_{R=1}^{r} \mu\_\text{rule}_R(E)} \tag{3}$$

where E is the new example, C_R^{out} the centre of the output fuzzy set of rule R, and r the total number of rules.

4 FUZZY RULE GENERATION USING COVERING ALGORITHMS

This section presents a new technique that allows a covering algorithm to generate fuzzy rules. This technique is especially adapted to algorithms based on the structure of RULES-5. RULES-5 iteratively employs a specific rule-forming process in order to create a complete rule set covering all examples. Three particular steps of this process are of interest to the development of RULES-F.

The first step in this process is the selection of a seed example (SE), the example used to generate a new rule. In RULES-5, SE is the first example in the list of training examples not covered by previously created rules. The second step employs a specific search process to create a consistent and general rule covering SE. The main characteristic of this search is that the ranges for numerical attributes are created automatically during the rule-forming process. They are not prediscretized. The result is a rule R, where all numerical conditions take the form $[Vmin_R^i < A^i < Vmax_R^i]$ (excluding the edges). These conditions might cover large areas in the example space. Thus, as the third and final step, the algorithm employs a post-processing technique that reduces the coverage of some numerical attribute conditions to the examples in T only. All numerical conditions will take the form $[Vmin_R^i \leqslant A^i \leqslant Vmax_R^i]$. This avoids the coverage of 'unknown' areas and reduces the possibility of overlapping rules.

This section describes modifications to each of these three steps to allow the creation of fuzzy rules.

4.1 Seed example selection in RULES-F

When employing RULES-F, the output values are numerical (V_E^{out}). However, to employ the RULES-5 rule-forming process, a target class has to be predefined. Thus, each output value has to be fuzzified in order to determine the target output fuzzy set before starting the creation of a rule. It is proposed to split the numerical output range into a fixed number (Nf, determined by the user) of membership functions. This number can be seen as a precision degree prescribed for the model; the higher the number, the higher the accuracy of the created rule set. However, this will also cause the number of rules to increase. Thus, the user has a degree of control over the size and precision of the model.

Given the number, Nf, of required membership functions and the output range $[V_{\min}^{\text{out}}, V_{\max}^{\text{out}}]$, the algorithm decomposes the output range into Nf triangular fuzzy sets $(F_1^{\text{out}}, \ldots, F_k^{\text{out}}, \ldots, F_{Nf}^{\text{out}})$, defined as

$$F_k^{\text{out}} = \text{Tr}(a(k), b(k), c(k))$$

where k is an integer $\in$ [1, Nf], $b(k) = (V^{\text{out}}_{\max} - V^{\text{out}}_{\min}) \cdot (k-1)/(\text{Nf}-1)$, $a(k) = b(k) - (V^{\text{out}}_{\max} - V^{\text{out}}_{\min})/(\text{Nf}-1)$, and $c(k) = b(k) + (V^{\text{out}}_{\max} - V^{\text{out}}_{\min})/(\text{Nf}-1)$.

For instance, Fig. 2 shows the fuzzification when $V^{\text{out}}_{\min} = 0$, $V^{\text{out}}_{\max} = 15$, and Nf = 4.

On the basis of this decomposition, each output value V^{out}_E could belong to two output fuzzy sets (F^{out}_k and F^{out}_{k+1}). In order to fuzzify V^{out}_E, only one of those sets is selected.

Thus, for a given output value V^{out}_E, the selected output fuzzy set will be the set F^{out}_k for which the membership degree $\mu_{F^{\text{out}}_k}(V^{\text{out}}_E)$ is greater; in other words, the fuzzy set F^{out}_k such that $V^{\text{out}}_E \in]b(k) - a(k))/2, (c(k) - b(k))/2[$ and $\mu_{F^{\text{out}}_k}(V^{\text{out}}_E) > 0.5$. In the particular case where the membership degree is equal for the two membership functions (F^{out}_k and F^{out}_{k+1}), that is, $V^{\text{out}}_E = (b(k+1) - a(k+1))/2 = (c(k) - b(k))/2$ and $\mu_{F^{\text{out}}_k}(V^{\text{out}}_E) = 0.5$, one fuzzy set will be selected randomly.

Now that the output fuzzy sets are defined, the algorithm can select a SE with its corresponding output fuzzy set in order to form a rule.

In RULES-5, as already mentioned, SE is selected among the examples not covered by the previously created rules. In the case of RULES-F, an example E might be covered by an existing rule R with its output value (V^{out}_E) belonging to F^{out}_R ($\mu_{F^{\text{out}}_R}(V^{\text{out}}_E) > 0$). However, there might be another output fuzzy set F^{out}_k, as defined earlier, where $\mu_{F^{\text{out}}_k}(V^{\text{out}}_E) > \mu_{F^{\text{out}}_R}(V^{\text{out}}_E)$. In such cases, the algorithm considers that F^{out}_R does not properly represent the example output value. Therefore, it needs to create another rule for this example.

Thus, in RULES-F, the SE selected is the first example in T that is not covered by at least one previously created rule R for which $\mu_{F^{\text{out}}_R}(V^{\text{out}}_{\text{SE}}) \geqslant 0.5$.

4.2 Formation of a rule

The RULES-5 rule-forming process has been designed for examples with nominal classes. Therefore, in order to be able to use this process, the output values of all examples (not just the seed example SE) in T need to be discretized.

During the selection of SE, the target fuzzy set has been identified (F^{out}_R). For the discretization of the output values of the examples, an example E is classified as a positive example (in the target class) if E has an output value belonging to F^{out}_R ($\mu_{F^{\text{out}}_R}(V^{\text{out}}_E) > 0$) and the remaining examples are taken as negative.

The rule-forming process of RULES-5 is then used unchanged. At the end of the rule formation process, a rule R is obtained to cover SE and other examples belonging to the positive class. Each condition in R takes the form $\text{Cond}^i_R = [A^i = V^i_{\text{SE}}]$ and $\text{Cond}^i_R = [Vmin^i_R < A^i < Vmax^i_R]$ for nominal and numerical attributes, respectively.

4.3 Rule post-processing (fuzzification of conditions)

In the created rules, the conditions for numerical attributes are formed with as large ranges as possible (Fig. 3). The post-processing (reduction of ranges) illustrated in Fig. 3, which is done in RULES-5 to avoid overlapping and coverage of unknown areas, is not necessary with RULES-F because the fuzzy logic representation permits the handling of such situations. However, other inductive learning post-processing operations such as pruning [**23**] may be used in order to reduce the sizes of the created rule sets when the training examples are noisy.

After the rule-forming process, the class of the rule (positive) is replaced by the target fuzzy set F^{out}_R and each condition is fuzzified using the following methods.

1. In the case of a numerical attribute, $\text{Cond}^i_R = [Vmin^i_R < A^i < Vmax^i_R]$ is transformed into the fuzzy condition $\text{Cond}^i_R = [A^i$ is Tr(a, b, c)], where $a = Vmin^i_R$, $b = (Vmax^i_R - Vmin^i_R)/2$, and $c = Vmax^i_R$, if $Vmin^i_R$ and $Vmax^i_R$ are finite; $a \to -\infty$, $b = V^i_{\min}$, and $c = Vmax^i_R$, if $Vmin^i_R \to -\infty$; $a = Vmin^i_R$, $b = V^i_{\max}$, and $c \to +\infty$, if $Vmax^i_R \to +\infty$.
2. In the case of a nominal attribute, the condition $\text{Cond}^i_R = [A^i = V^i_{\text{SE}}]$ is transformed into the fuzzy condition $\text{Cond}^i_R = [A^i$ is $\text{Vcod}^i_k]$, where Vcod^i_k is the coded value of V^i_{SE}.

The adoption of the RULES-5 rule-forming process enables the algorithm to create membership

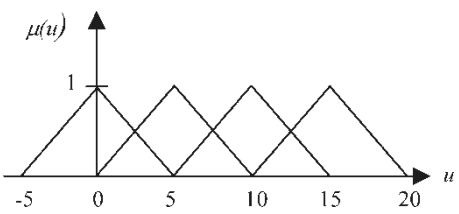


Fig. 2 Output fuzzification

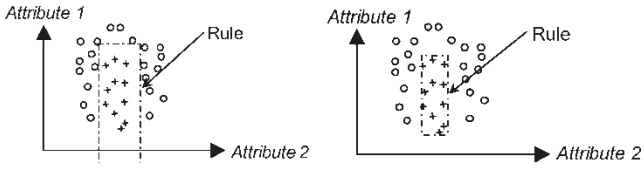


Fig. 3 Conditions with large and reduced ranges

functions for each numerical attribute during the generation of each rule. In addition to being an automatic process, this method allows the use of a wider range of fuzzy sets in the same rule set than possible with other fuzzification techniques. Thus, for the same attribute, different membership functions with different widths are generated for each rule. The algorithm itself determines this width with a view to optimizing the coverage of a rule.

5 ILLUSTRATIVE PROBLEM

To illustrate the process, a simple example is used. The example involves a training set T containing 250 examples (Fig. 4). An example in T comprised a nominal attribute Curve-type (1, 2), a numerical attribute Ax, and a numerical output Ay. The following mathematical model represents the example space (Fig. 5).

IF Curve-type $= 1$ THEN Ay $=$ sin (Ax)
IF Curve-type $= 2$ THEN
  IF Ax $= k \cdot \pi$ THEN Ay $= 0$
  IF Ax $\in]\, 2 \cdot k \cdot \pi, (2k+1) \cdot \pi\, [$THEN Ay $= 1$
  IF Ax $\in]\, (2 \cdot k+1) \cdot \pi, (2 \cdot k+2) \cdot \pi\, [$THEN
    Ay $= -1$

where k is an integer $\in [0,+\infty[$

Assuming that this mathematical model is too complex to be found, based on T, the RULES-F algorithm can be used to develop a predictive model. The precision for the model is fixed at Nf $= 4$ membership functions for the output. The four possible output membership functions F1, F2, F3, and F4 are created (Fig. 6). All rules will use one of these membership functions for the output fuzzy set and the RULES-F process will follow the steps described below.

	Curve-type	Ax	Ay
1	1	0.1	0.099833
2	2	0.1	1
3	1	0.2	0.198669
4	2	0.2	1
5	1	0.3	0.29552
6	2	0.3	1
7	1	0.4	0.389418
8	2	0.4	1
...	...	...	...
67	1	3.4	-0.25554
68	2	3.4	-1
69	1	3.5	-0.35078
70	2	3.5	-1
...	...	...	...
249	1	12.5	-0.06632
250	2	12.5	-1

Fig. 4 Example set

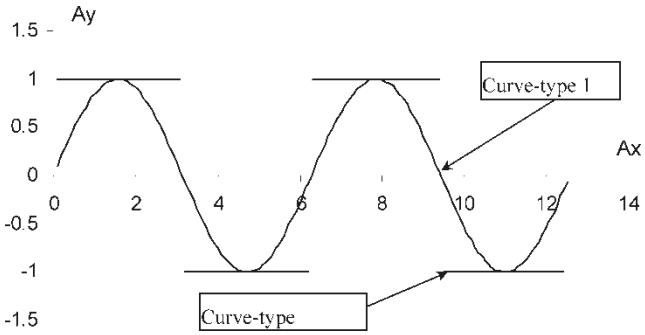


Fig. 5 Representation of the example set

Selection of a seed example. In order to form the first rule, the first example is taken as the seed example

$$SE = (\text{Curve-type} = 1, Ax = 0.1, Ay = 0.099833)$$

The target output fuzzy set that maximizes the membership degree of the seed example is F3.

Rule formation. In order to use the RULES-5 rule-forming process, the output values of the examples are discretized using the target fuzzy set. If the output value of an example belongs to F3, it is classified as positive otherwise it is labelled as negative. The resulting training set is shown in Fig. 7.

The RULES-5 rule-forming process is then applied to this training set using the following SE

$$SE = (\text{Curve-type} = 1, Ax = 0.1, \text{class} = \text{positive})$$

The result of the process is the following rule

$$\text{best\_rule} = \text{IF}[\text{Curve-type} = 1] \text{ AND } [Ax < 3.5], \text{ THEN class} = \text{positive}$$

Rule fuzzification. The fuzzification of best_rule starts. The class (positive) is replaced by the target fuzzy set F3, the condition [Ax < 3.5] is transformed into the fuzzy condition [Ax is Tr($-\infty$, 0.1, 3.5)] and the nominal value Curve-type $= 1$ is fuzzified using the coded value 1 to form a singleton fuzzy set (as explained in section 3) and the condition [Curve-type $= 1$] is transformed into the fuzzy condition [Curve-type is 1]. The following rule is obtained

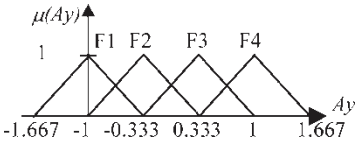


Fig. 6 Fuzzification of Ay with Nf $= 4$

	Curve-type	Ax	Ay
1	1	0.1	positive
2	2	0.1	negative
3	1	0.2	positive
4	2	0.2	negative
5	1	0.3	positive
6	2	0.3	negative
7	1	0.4	positive
8	2	0.4	negative
...	...	...	...
67	1	3.4	positive
68	2	3.4	negative
69	1	3.5	negative
70	2	3.5	negative
...	...	...	...
249	1	12.5	positive
250	2	12.5	negative

Fig. 7 Discretized example set

and added to the rule set

IF[Curve-type is 1] AND [Ax is Tr($-\infty$, 0.1, 3.5)], THEN Ay is F3

As there are still examples not covered by the rule set, the algorithm selects another SE. For instance

SE = (Curve-type = 2, Ax = 0.1, Ay = 1)

When applying the procedure to this example, the result is the following rule

IF[Curve-type is 2] AND [Ax is Tr($-\infty$, 0.1, 3.2)], THEN Ay is F4

At the end of the entire process, the result is the 12 rules shown in Fig. 8. The curves produced from this model can be compared with the original curves in Fig. 9.

Using Nf = 10, i.e. adopting ten membership functions for the output, the result of the algorithm would be 36 rules. The curves from this model can be compared with the original curves in Fig. 10. This clearly shows the influence of Nf. As mentioned previously, high values create more accurate models but also larger rule sets. It is of note that some areas of the attribute space require more precision than others (smaller output membership functions). Thus, the automatic creation or refinement of output membership functions could be advantageous.

R1 IF [Curve-type is 1] AND [Ax is *Tr* (-∞, 0.1, 3.5)] THEN Ay is F3
R2 IF [Ax is *Tr* (0.3, 1.6, 2.9)] THEN Ay is F4
R3 IF [Curve-type is 1] AND [Ax is *Tr* (2.8, 4.75, 6.7)] THEN Ay is F2
R4 IF [Ax is *Tr* (3.4, 4.7, 6.)] THEN Ay is F1
R5 IF [Curve-type is 1] AND [Ax is *Tr* (5.9, 7.9, 9.8)] THEN Ay is F3
R6 IF [Ax is *Tr* (6.6, 7.9, 9.1)] THEN Ay is F3
R7 IF [Curve-type is 1] AND [Ax is *Tr* (9,12.5,+ ∞)] THEN Ay is F2
R8 IF [Ax is *Tr* (9.7, 11, 12.3)] THEN Ay is F1
R9 IF [Curve-type is 2] AND [Ax is *Tr* (-∞,0.1, 3.2)] THEN Ay is F4
R10 IF [Curve-type is 2] AND [Ax is *Tr* (3.1, 4.7, 6.3)] THEN Ay is F1
R11 IF [Curve-type is 2] AND [Ax is *Tr* (6.2, 7.9, 9.5)] THEN Ay is F4
R12 IF [Curve-type is 2] AND [Ax is *Tr* (9.4,12.5, · ∞)] THEN Ay is F1

Fig. 8 Rule set obtained for Nf = 4

6 TESTS AND ANALYSIS OF RESULTS

Because of its simplicity and good performance, Wang and Mendel's algorithm is still widely used, especially for control applications. A comparison between RULES-F and that algorithm has been carried out. Two practical cases have been used. The first problem is part of a research project in progress at the authors' centre, which aims to develop a model for the guidance of a robotic arm. The second problem concerns the control of a truck and was also employed by Wang and Mendel to evaluate their algorithm [**10**]. To assess the performance of each created model, a graphical representation of the deviation between the actual output and the predicted output will be shown and the following measures will be used.

1. Maximum absolute error = maximum absolute deviation between the actual output values and the values predicted by the evaluated model obtained with the examples in T.
2. Mean absolute error = average of the absolute deviations obtained with the examples in T.

6.1 Fuzzy model for robot arm control

The problem involved in the creation of a fuzzy input–output dynamic model for the control of a PUMA 560 robot [**24**, **25**]. The position (X, Y, Z) of the robot end-effector depends on three joints and can be defined by the angles at these joints ($\theta 1$, $\theta 2$, $\theta 3$). T contains 27 825 examples. An example in T comprised six input attributes, the joint angles at time t ($\theta 1_t$, $\theta 2_t$, $\theta 3_t$) and at time $t-1$ ($\theta 1_{t-1}$, $\theta 2_{t-1}$, $\theta 3_{t-1}$), and three outputs, the resulting spatial position (X_{t+1}, Y_{t+1}, Z_{t+1}). The assumption underlying the model is that X_{t+1}, Y_{t+1} and Z_{t+1}, which are functions of the joint angles $\theta 1_{t+1}$, $\theta 2_{t+1}$ and $\theta 3_{t+1}$ at time $t+1$, can be expressed in general terms as

$$X_{t+1} = f(\theta 1_t, \theta 2_t, \theta 3_t, \theta 1_{t-1}, \theta 2_{t-1}, \theta 3_{t-1}) \quad (4)$$

$$Y_{t+1} = g(\theta 1_t, \theta 2_t, \theta 3_t, \theta 1_{t-1}, \theta 2_{t-1}, \theta 3_{t-1}) \quad (5)$$

$$Z_{t+1} = h(\theta 1_t, \theta 2_t, \theta 3_t, \theta 1_{t-1}, \theta 2_{t-1}, \theta 3_{t-1}) \quad (6)$$

Building a model consists of presenting the example septuplets ($\theta 1_t$, $\theta 2_t$, $\theta 3_t$, $\theta 1_{t-1}$, $\theta 2_{t-1}$, $\theta 3_{t-1}$, X_{t+1}), ($\theta 1_t$, $\theta 2_t$, $\theta 3_t$, $\theta 1_{t-1}$, $\theta 2_{t-1}$, $\theta 3_{t-1}$, Y_{t+1}), and ($\theta 1_t$, $\theta 2_t$, $\theta 3_t$, $\theta 1_{t-1}$, $\theta 2_{t-1}$, $\theta 3_{t-1}$, Z_{t+1}) taken from the

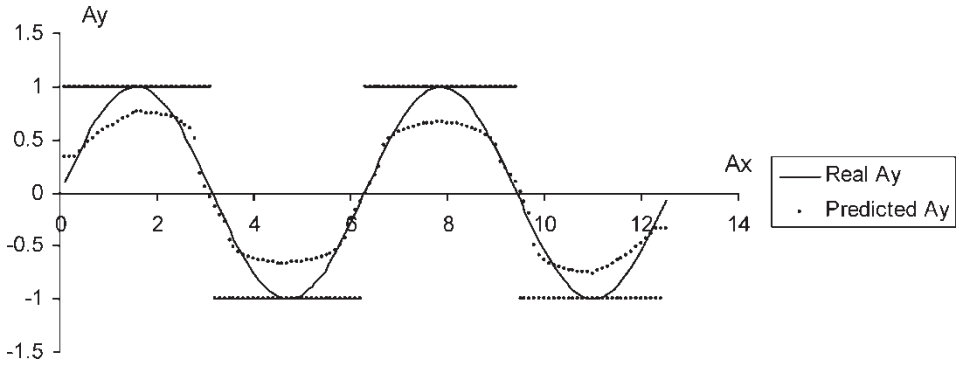


Fig. 9 Prediction obtained for Nf = 4

training set *T* to a machine-learning algorithm for it to induce the functions *f*, *g*, and *h*.

A model was first created by applying Wang and Mendel's algorithm. The membership functions had to be predefined for all input attributes and outputs. The universes of discourse of the six attributes and the three outputs were all decomposed into ten triangular membership functions. The resulting model comprises 389 rules for predicting the three outputs. The predictions of the model for the first 10 000 examples compared with the actual outputs *X*, *Y*, and *Z* are shown, respectively, in Figs 11(a), 12(a), and 13(a).

1. For output *X*, maximum absolute error equals 0.111469 and mean absolute error equals 0.030012.
2. For output *Y*, maximum absolute error equals 0.138182 and mean absolute error equals 0.026793.
3. For output *Z*, maximum absolute error equals 0.115952 and mean absolute error equals 0.020622.

Using RULES-F, another model was then created. Because RULES-F can only handle one output at a time, three training sets were generated in order to produce one rule set for each output. The first advantage when employing RULES-F is that only the universes of discourse of the three outputs need to be decomposed into membership functions.

As had been done in Wang and Mendel's algorithm, the number of membership functions was fixed at ten. Parameter PRSET_size in the RULES-5 rule-forming process [**9**] was set to 2 and no pruning process was employed. The result was 67 rules for the prediction of *X*, 33 rules for the prediction of *Y*, and 69 rules for the prediction of *Z*, a total of 169 rules. The predictions of the model for the first 10 000 examples compared with the actual outputs *X*, *Y*, and Z are shown, respectively, in Figs 11(b), 12(b), and 13(b).

1. For output *X* maximum absolute error equals 0.089994 and mean absolute error equals 0.019921.
2. For output *Y* maximum absolute error equals 0.088321 and mean absolute error equals 0.015609.
3. For output *Z* maximum absolute error equals 0.044578 and mean absolute error equals 0.004518.

RULES-F required a much longer execution time than Wang and Mendel's algorithm (∼10 min against a few seconds on a Pentium IV 1 GHz computer), but it created a much more compact rule set, 56 per cent smaller, which is also more accurate, with a mean absolute error lower by 66 per cent. The differences

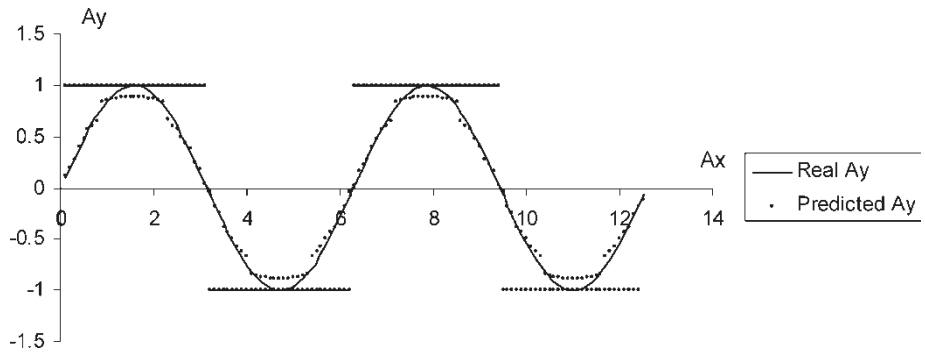
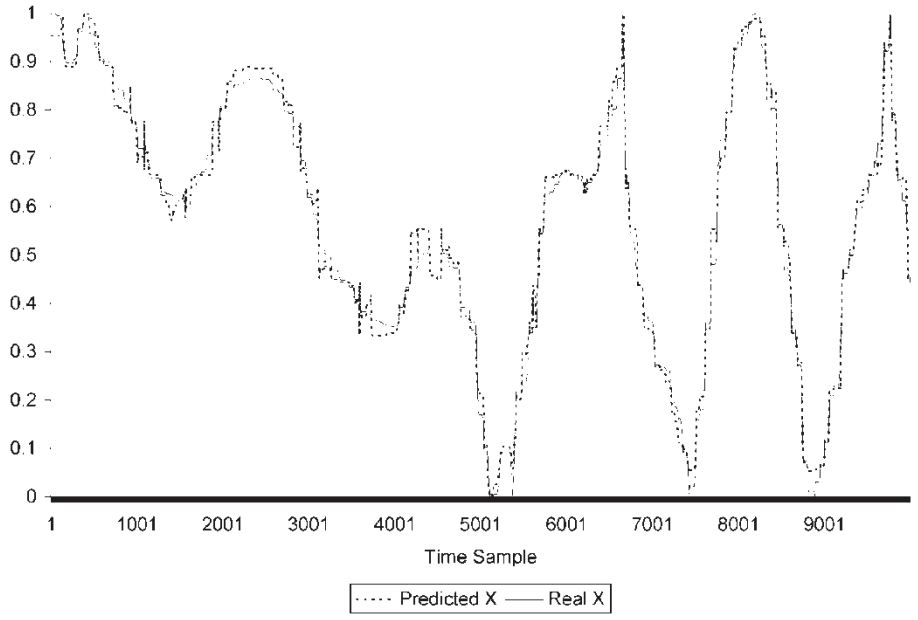


Fig. 10 Prediction obtained for Nf = 10

(a) Results for Wang and Mendel's algorithm



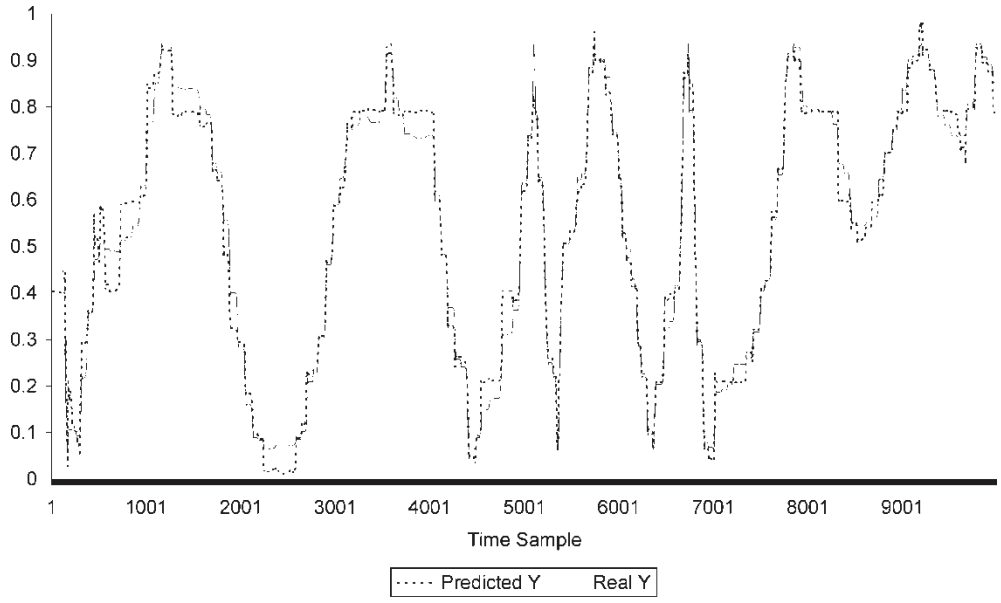
(b) Results for RULES-F

Fig. 11 Prediction of output X

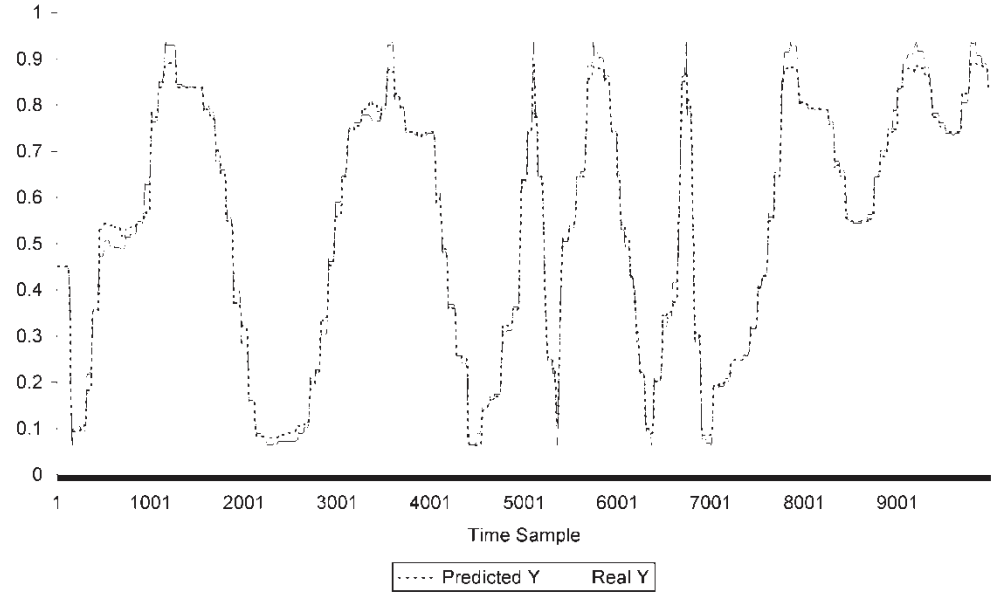
in accuracy are clearly shown in the graphical representations.

6.2 Fuzzy model for truck control

The aim of this experiment was to create a model for the control of a truck reversing to a specified loading dock. Each example in T represents one position of the truck (Fig. 14), defined by the angle (φ) of the truck relative to the horizontal axis and the location (x) of the truck on the horizontal axis, with the resulting required action on the steering wheel, defined by the required steering angle (θ). Thus, an example is composed of two input attributes (φ and x) and one output (θ). T contains 238 examples.



(a) Results for Wang and Mendel's algorithm

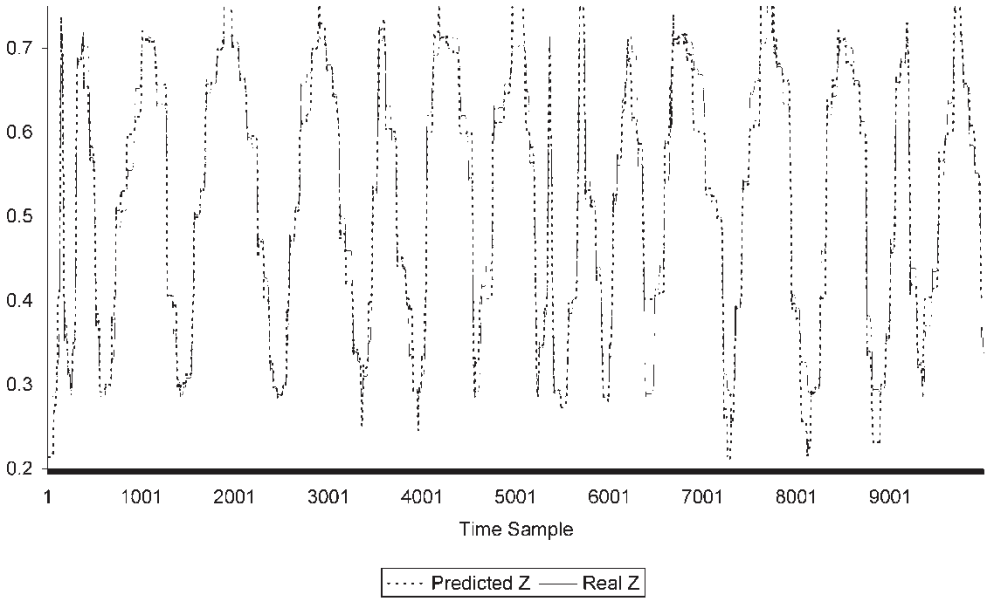


(b) Results for RULES-F

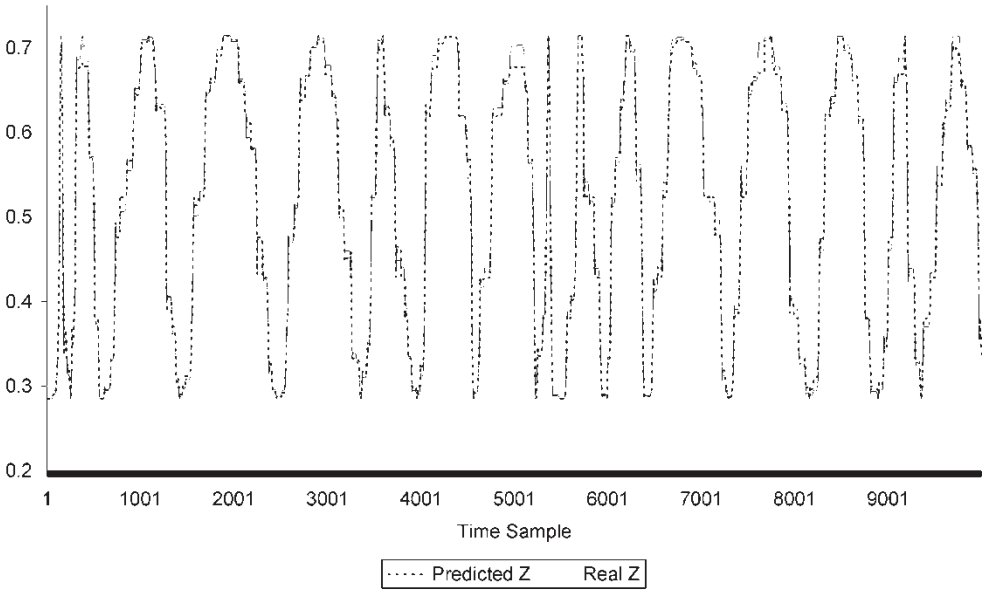
Fig. 12 Prediction of output Y

After having predefined the membership functions for the input attributes and output, Wang and Mendel applied their algorithm and obtained 27 rules. The prediction of the model for the 238 examples compared with the actual output θ is shown in Fig. 15(a). For this model, the maximum absolute error equals 35 and the mean absolute error equals 6.676.

In the case of RULES-F, only the output universe of discourse needed to be decomposed into membership functions. Similar to the case of Wang and Mendel's experiment, the output was decomposed into seven membership functions (Fig. 16). Note that the decomposition performed by Wang and Mendel is slightly different from that achieved with RULES-F, because the current implementation of



(a) Results for Wang and Mendel's algorithm



(b) Results for RULES-F

Fig. 13 Prediction of output Z

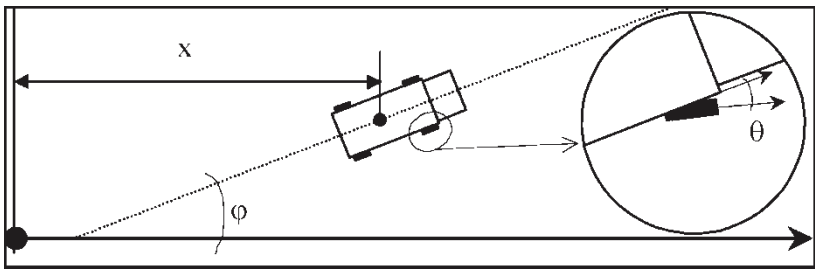
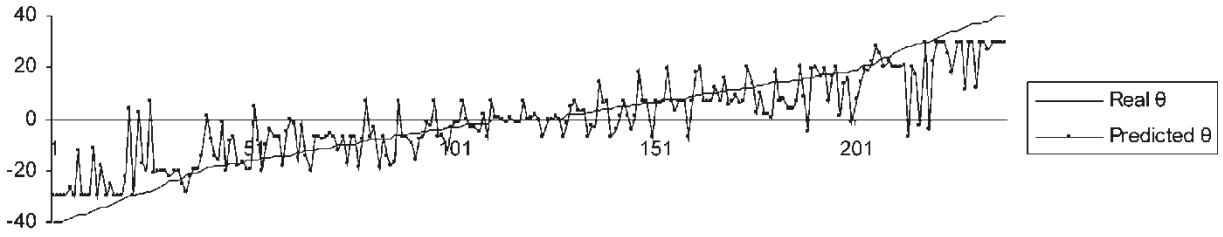
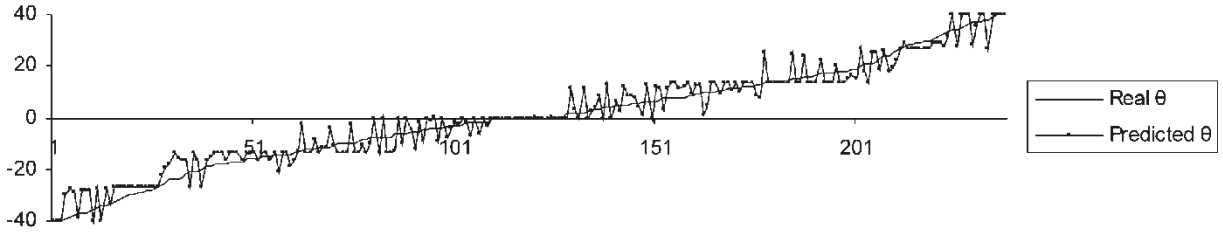


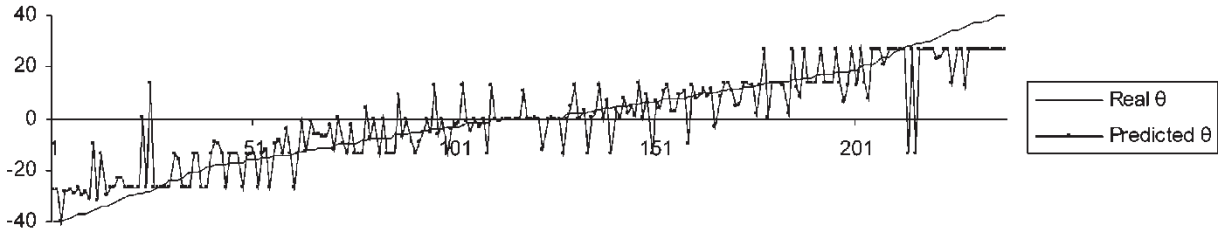
Fig. 14 Truck control problem



(a) Wang and Mendel's algorithm results



(b) RULES-F with NL = 0 results



(c) RULES-F withNL = 0.25 results

Fig. 15 Prediction of output θ

RULES-F deals only with decomposition (for the output) into membership functions of equal bases. Allowing the user to specify membership functions or automatically constructing them increases flexibility and could yield improved results.

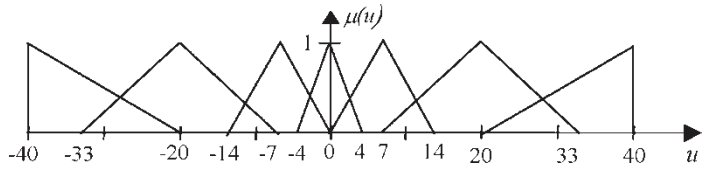
RULES-F was applied twice, once with and once without the incremental post-pruning (IPP) process [**23**]. For both cases, the RULES-5 specific parameter PRSET_size was fixed to 2.

The output of RULES-F, without pruning, is a rule set containing 70 rules. The prediction of the model for the 238 examples compared with the actual output θ is shown in Fig. 15(b). For this model, the maximum absolute error equals 11.764 and the mean absolute error equals 3.484.

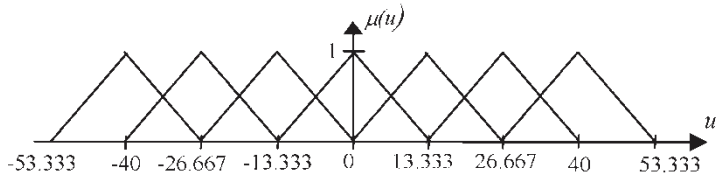
The output of RULES-F, using the IPP process (with a noise level NL equals 0.25), is a rule set containing 17 rules (Fig. 17). The prediction of the model for the 238 examples compared with the actual output θ is shown in Fig. 15(c). For this model, the maximum absolute error equals 29 and the mean absolute error equals 6.35.

Thus, without pruning, RULES-F produced an accurate rule set, with a mean absolute error reduced by 47.8 per cent compared with that of Wang and Mendel's algorithm, but it also generates more rules.

However, if the size of the rule set is a problem, as illustrated in this case, RULES-F can be combined with pruning to remove rules created because of noisy examples. The use of IPP permits the creation of a more compact rule set, albeit with a reduction in accuracy considered to be due to the presence of noisy examples. For the case illustrated, this rule



Decomposition used by Wang and Mendel's algorithm



Decomposition used by DynaFuzz

Fig. 16 Output membership functions

R1	If [φ is Tr(85.05,114.08,143.11)]&[x is Tr(9.8,12.18,14.55)]	then θ is Tr(-13.33,0,13.33)
R2	If [φ is Tr(41.78,66.42,91.06)]&[x is Tr(4.65,7.335,10.02)]	then θ is Tr(-13.33,0,13.33)
R3	If [φ is Tr(18.75,53.95,89.15)]&[x is Tr(5.96,9.275,12.59)]	then θ is Tr(-26.67,-13.34,0)
R4	If [φ is Tr(90.56,270,+∞)]&[x is Tr(9.81,11.02,12.23)]	then θ is Tr(0,13.34,26.67)
R5	If [φ is Tr(134.69,202.35,270)]	then θ is Tr(13.33,26.67,40)
R6	If [φ is Tr(-90,-22.35,45.31)]	then θ is Tr(-40,-26.67,-13.33)
R7	If [φ is Tr(110.77,124.5,138.22)]&[x is Tr(11.28,19 ,+∞)]	then θ is Tr(-13.33,0,13.33)
R8	If [φ is Tr(0,41.58,83.16)]&[x is Tr(9.58,19,+∞)]	then θ is Tr(-40,-26.67,-13.33)
R9	If [φ is Tr(97.5,270,+∞)]&[x is Tr(-∞,1,10.32)]	then θ is Tr(13.33,26.67,40)
R10	If [φ is Tr(131.69,270,+∞)]&[x is Tr(-∞,1,13.46)]	then θ is Tr(26.67,40,53.33)
R11	If [φ is Tr(66.9,100.795,134.69)] &[x is Tr(13.83,19,+∞)]	then θ is Tr(-26.67,-13.34,0)
R12	If [φ is Tr(54.91,78.145,101.38)] &[x is Tr(-∞,1,7.71)]	then θ is Tr(0,13.34,26.67)
R13	If [φ is Tr(-∞,-90,44.04))]&[x is Tr(6.54,19,+∞)]	then θ is Tr(-53.33,40,-26.67)
R14	If [φ is Tr(26.59,270,+∞)]&[x is Tr(-∞,1,4.27)]	then θ is Tr(0,13.34,26.67)
R15	If [φ is Tr(128.7,270,+∞)]&[x is Tr(15.84,19,+∞)]	then θ is Tr(13.33,26.67,40)
R16	If [φ is Tr(-∞,-90,51.3)]&[x is Tr(-∞,1,4.16)]	then θ is Tr(-40,-26.67,-13.33)
R17	If [φ is Tr(86.91,94.145,101.38)]&[x is Tr(-∞,1,9.81)]	then θ is Tr(0,13.34,26.67)

Fig. 17 Rules created by RULES-F with NL = 0.25

set is more compact and still more accurate than Wang and Mendel's algorithm. Figure 18 shows the plots of deviations between the predicted and actual θ values. These plots further demonstrate the better performance of RULES-F models, in particular, in the case where NL = 0.

7 CONCLUSION

This paper has presented a new method for fuzzy rule induction based on a modification of the RULES-5 inductive learning algorithm to enable the creation of models for the prediction of numerical

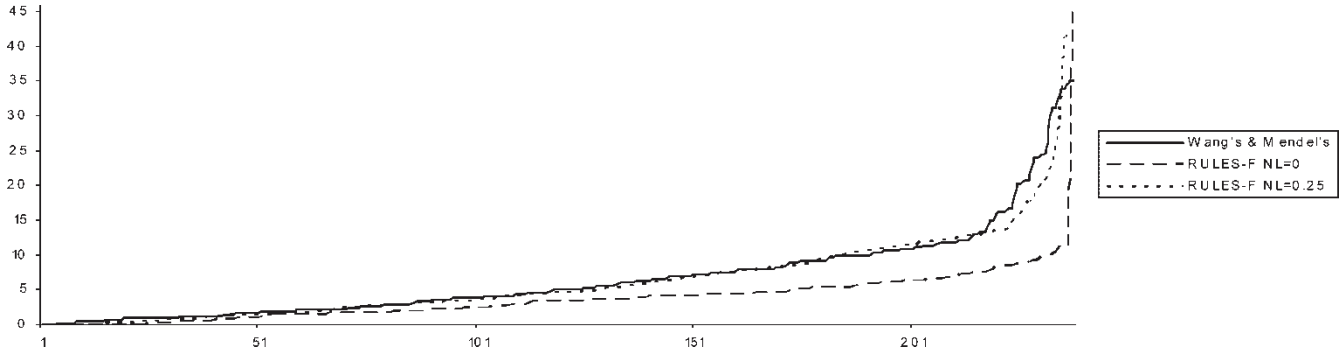


Fig. 18 Prediction deviations from real θ

outputs. The resulting algorithm, called RULES-F, combines the capabilities of fuzzy logic for numerical output and uncertainty handling with the good performance of RULES-5. Tests have shown that compared with a well-known algorithm for fuzzy rule induction, RULES-F permits the creation of more compact and more accurate fuzzy rule sets. Thus, this research has yielded an algorithm that seems appropriate for control applications, which is a departure from the typical applications of inductive learning algorithms to date. Further tests should be carried out in this application domain to confirm the good performance of RULES-F. It is likely that RULES-F will perform better than Wang and Mendel's algorithm on most problems, as it uses more effective machine-learning techniques. In this respect, the results obtained should also be compared with those produced by other machine-learning -based methods, such as neuro-fuzzy algorithms. Finally, tests have shown that the performance of the algorithm is highly dependent on the number of output membership functions used. Further research could be conducted to automate the creation or refinement of output membership functions. In contrast, the new fuzzy induction algorithm could also be modified to allow users to define their own input membership functions if so desired.

ACKNOWLEDGEMENT

This work was carried out within the ERDF (Objective 1) project 'Supporting Innovative Product Engineering and Responsive Manufacturing' (SUPERMAN-2) and the EC FP6 I*PROMS project.

REFERENCES

1 **Quinlan, J. R.** Learning with continuous classes. Proceedings of the 5th Australian Joint Conference on *Artificial intelligence*, World Scientific, Singapore, 1992, pp. 343–348.

2 **Breiman, L., Friedman, J. H., Olshen, R. A.,** and **Stone, C. J.** *Classification and regression trees*, 1984 (Wadsworth & Brooks/Cole Advanced Books & Software, Belmont, CA, USA).

3 **Morimoto, Y., Ishii, H.,** and **Morishita, S.** Efficient construction of regression trees with range and region splitting. Proceedings of the 23rd Very Large Data Bases Conference, Athens, Greece, 1997, pp. 166–175.

4 **Karalic, A.** and **Cestnik, B.** The Bayesian approach to tree-structured regression. Proceedings of Information Technology Interfaces '91, Cavtat, Yugoslavia, 1991, pp. 155–160.

5 **Bruha, I.** and **Berka, P.** Continuous classes in rule induction: empirical comparison of two approaches. 3rd International Workshop on *Artificial intelligence techniques*, Brno, Czech Republic, 1996, pp. 163–164.

6 **Karalic, A.** Employing linear regression in regression tree leaves. Proceedings of European Conference on *Artificial intelligence '92*, Vienna, Austria, 1992, pp. 440–441.

7 **Sebag, M.** and **Schoenauer, M.** Inductive learning of membership functions and fuzzy rules. *Uncertainty modeling: theory and applications. Machine intelligence and pattern recognition series*, 1994, pp. 275–283 (North Holland, Netherlands).

8 **Nozaki, K.** and **Ishibuchi, H.** A simple but powerful heuristic method for generating fuzzy rules from numerical data. *Fuzzy Sets Syst.*, 1997, **86**, 251–270, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

9 **Pham, D. T., Bigot, S.,** and **Dimov, S. S.** RULES-5: a rule induction algorithm for classification problems involving continuous attributes. *Proc. Instn Mech. Engrs, Part C: J. Mechanical Engineering Science*, 2004, **217**, 1273–1286.

10 **Wang, L. X.** and **Mendel, J. M.** *Generating fuzzy rules from numerical data, with applications.* Technical Report TR USC-SIPI #169, Signal and Image Processing Institute, University of Southern California, CA, USA, 1991.

11 **Delgado, M.** and **Gonzalez, A.** An inductive learning procedure to identify fuzzy systems. *Fuzzy Sets Syst.*, 1993, **55**, 121–132, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

12 **Hong, T. P.** and **Lee, C. Y.** Induction of fuzzy rules and membership functions from training examples. *Fuzzy Sets Syst.*, 1996, **84**, 33–47, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

13 **Hong, T. P.** and **Lee, C. Y.** Effect of merging order on performance of fuzzy induction. *Intell. Data Anal.*, 1999, **3**(2), 139–151.

14 **Hong, T. P.** and **Chen, J. B.** Processing individual fuzzy attributes for fuzzy rule induction. *Fuzzy Sets Syst.*, 2000, **112**, 127–140, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

15 **Shann, J. J.** and **Fu, H. C.** A fuzzy neural network for rule acquiring on fuzzy control systems. *Fuzzy Sets Syst.*, 1995, **71**, 345–357, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

16 **Jeng, B., Jeng, Y. M.,** and **Liang, T. P.** FILM: a fuzzy inductive learning method for automated knowledge acquisition. *Deci. Support Syst.*, 1997, **21**(2), 61–74.

17 **Quinlan, J. R.** Induction of decision trees. *Mach. Learn.*, 1986, **1**, 81–106.

18 **Castro, J. L., Castro-Schez, J. J.,** and **Zurita, J. M.** Use of a fuzzy machine learning technique in the knowledge acquisition process. *Fuzzy Sets Syst.*, 2001, **123**, 307–320, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

19 **Wang, C. H., Liu, J. F., Hong, T. P.,** and **Tseng, S. S.** A fuzzy inductive learning strategy for modular rules. *Fuzzy Sets Syst.*, 1999, **103**, 91–105, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

20 **Ravi, V., Zimmermann, H. J.,** and **Reddy, P. J.** Fuzzy rule base generation for classification and its minimization via modified threshold accepting. *Fuzzy Sets Syst.*, 2001, **120**, 271–279, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

21 **Wang, X. Z., Wang, Y. D., Xu, X. F., Ling, W. D.,** and **Yeung, D. S.** A new approach to fuzzy rule generation: fuzzy extension matrix. *Fuzzy Sets Syst.*, 2001, **123**, 291–306, available from http://www.elsevier.com/locate/fss (accessed 2 August 2004).

22 **Rondeau, L., Ruelas, R., Levrat, L.,** and **Lamotte, M.** A defuzzification method respecting the fuzzification. *Fuzzy Sets Syst.*, 1997, **86**, 311–320, available from http://www.elsevier.com/locate/fss (accessed 3 October 2002).

23 **Pham, D. T., Bigot, S.,** and **Dimov S. S.** RULES-5: a rule merging technique for handling noise in inductive learning. *Proc. Instn Mech. Engrs, Part C: J. Mechanical Engineering Science*, 2003, **217**, 1273–1286.

24 **Armstrong, B.** and **Khatib, O.** The explicit dynamic model and inertial parameters of the PUMA 560 Robot arm. International Conference on *Robotics and automation*, San Francisco, CA, USA, 1986, Vol. **1**, pp. 510–518.

25 **Craig, J. J.** *Introduction to robotics: mechanics and control*, 3rd edition, 2004 (Prentice Hall, Englewood Cliffs, NJ).

APPENDIX 1

A^i	ith attribute in an example
Best_rule	best rule created by RULES-5
C_R^{out}	centre of the output fuzzy set of rule R
Cond_R^i	condition in rule R for the ith attribute
E	an example in T
F_R^i	fuzzy set employed in fuzzy rule R to form a condition on the ith attribute
F_R^{out}	output fuzzy set of rule R
F_k^{out}	kth fuzzy set created for the fuzzification of the output values, Tr($a(k)$, $b(k)$, and $c(k)$)
m	number of attributes in an example
Nf	number of output fuzzy sets fixed by the user
NL	noise level, parameter used in IPP process
PRSET_size	size of the partial rule set, a parameter used in RULES-5
R	a rule
r	number of rules in the rule set
SE	a seed example
T	a set of training examples
t^i	test value in a regression-tree model
Tr	triangular membership function
V_E^i	value of the ith attribute in example E
$V_{\max}^i$	maximum known value of the ith numerical attribute
$V_{\min}^i$	minimum known value of the ith numerical attribute
V_E^{out}	value of the output in example E
$V_{\max}^{\text{out}}$	maximum known value of the numerical output
$V_{\min}^{\text{out}}$	minimum known value of the numerical output
Vcod_k^i	coded value used as a fuzzy set for the fuzzification of a nominal value
Vd_k^i	kth nominal value of the ith attribute
$Vmax_R^i$	upper bound employed in rule R to form a condition on the ith numerical attribute
$Vmin_R^i$	lower bound employed in rule R to form a condition on the ith numerical attribute
$\mu\_\text{rule}_R(E)$	degree of match of example E with rule R
$\mu_F(u)$	degree of membership of value u of fuzzy set F

APPENDIX 2

An overview of RULES-5

RULES-5 employs simple and efficient techniques for extracting IF–THEN rules from examples. Data are presented to RULES-5 in the form of a collection of objects, each belonging to one of a number of given classes. These objects together with their associated classes constitute a set of training examples T from which the algorithm induces a model. Each example is described by its class value and by a vector of m attributes $(A_1, \ldots, A^i, \ldots, A^m)$. The algorithm employs a specialization process that searches for rules that are as general as possible and that correctly classifies all the training examples. Each rule is described by a conjunction of conditions on each attribute and by a target class value. The complete rule-forming procedure of RULES-5 can be summarized by the following procedure.

1. While there is an example in T not covered by any rule in the rule set formed so far:
   (a) select one of these uncovered examples, the seed example (SE);
   (b) Form the all inclusive rule covering SE: (IF no condition, THEN class is a class of SE);
2. While this rule covers example not belonging to SE class:
   (a) append a condition to it that excludes the closest example to SE that does not belong to SE class.
3. Add this rule to the rule set.

For more details on RULES-5, see reference [**9**].