Soft Comput (2011) 15:1959–1980
DOI 10.1007/s00500-010-0628-5

HILK++: an interpretability-guided fuzzy modeling methodology for learning readable and comprehensible fuzzy rule-based classifiers

José M. Alonso · Luis Magdalena

Published online: 22 June 2010

Abstract This work presents a methodology for building interpretable fuzzy systems for classification problems. We consider interpretability from two points of view: (1) readability of the system description and (2) comprehensibility of the system behavior explanations. The fuzzy modeling methodology named as Highly Interpretable Linguistic Knowledge (HILK) is upgraded. Firstly, a feature selection procedure based on crisp decision trees is carried out. Secondly, several strong fuzzy partitions are automatically generated from experimental data for all the selected inputs. For each input, all partitions are compared and the best one according to data distribution is selected. Thirdly, a set of linguistic rules are defined combining the previously generated linguistic variables. Then, a linguistic simplification procedure guided by a novel interpretability index is applied to get a more compact and general set of rules with a minimum loss of accuracy. Finally, partition tuning based on two efficient search strategies increases the system accuracy while preserving the high interpretability. Results obtained in several benchmark classification problems are encouraging because they show the ability of the new methodology for generating highly interpretable fuzzy rule-based classifiers while yielding accuracy comparable to that achieved by other methods like neural networks and C4.5. The best configuration of HILK will depend on each specific problem under consideration but it is important to remark that HILK is flexible enough (thanks to the combination of several algorithms in each modeling stage) to be easily adaptable to a wide range of problems.

Keywords Fuzzy modeling · Interpretability · Classification · Simplification · Tuning

J. M. Alonso (✉) · L. Magdalena
European Centre for Soft Computing (ECSC), 33600 Mieres, Asturias, Spain
e-mail: jose.alonso@softcomputing.es

L. Magdalena
e-mail: luis.magdalena@softcomputing.es

1 Introduction

Interpretable intelligent systems are always desired for all kind of applications (medicine, economics, robotics, etc.). Interpretability is really appreciated and it even becomes a strong requirement when dealing with humanistic systems, defined as *those systems whose behavior is strongly influenced by human judgment, perception or emotions* (Zadeh 1975).

This paper focuses on classification problems where interpretability is of prime concern. Of course, accuracy cannot be neglected because, at least at a given level, it is a prerequisite since a system which is not able to achieve a minimum accuracy is useless. Nevertheless, some applications can tolerate a minimum loss of accuracy if it means getting a transparent and comprehensible model. Sometimes, both criteria (accuracy and interpretability) can be satisfied to a high degree, but usually it is not possible because they somehow represent conflicting goals. Thus, looking for a good trade-off between them is one of the most difficult and challenging tasks in system modeling.

Interpretability is widely admitted to be the most valuable property of fuzzy rule-based systems (FRBSs). They are considered as gray boxes against other techniques such as neural networks which are viewed as black boxes. Fuzzy logic (FL) is acknowledged for its well-known ability for linguistic concept modeling mainly due to its semantic expressivity close to expert natural language, using linguistic variables (Zadeh 1975) and linguistic rules (Mamdani 1977). In consequence, FL represents a useful tool to tackle with the problem of building interpretable systems. In addition, it is especially useful to handle the intrinsic uncertainty of real-world problems where the available information is usually vague.

Moreover, being universal approximators (Castro 1995) FRBSs are able to perform nonlinear mappings between inputs and outputs. Thus, as explained by Hüllermeier (2005) there are lots of fuzzy machine learning methods for knowledge induction from experimental data.

On the other hand, the combination of several heterogeneous sources of knowledge (mainly expert and induced knowledge) is likely to yield compact and robust systems as pointed out by Alonso et al. (2008).

Notice that, the use of FL favors the interpretability of the final model but it is not enough to guarantee it (Alonso et al. 2009). Two main aspects must be taken into consideration when regarding interpretability of FRBSs (*Description* and *Explanation*). On the one hand, the system description has to be transparent enough to present the system as a whole describing its global behavior and trend. On the other hand, system explanation must consider all possible individual situations, explaining specific behaviors for specific events. Thus, comprehensibility of a FRBS depends on all its components, i.e., it depends on the knowledge base (including both variables and rules) transparency but also on the inference mechanism understanding.

Main aspects affecting to the readability of fuzzy systems have been thoroughly analyzed (Guillaume 2001). In addition, a complete study on the interpretability constraints most frequently used in fuzzy modeling has been recently published (Mencar and Fanelli 2008). Finally, in the fuzzy modeling literature there are two main trends regarding the search of the optimum interpretability–accuracy trade-off: (1) those first focused on interpretability and then on accuracy (Casillas et al. 2003a); (2) those who give priority to accuracy and then try to improve interpretability (Casillas et al. 2003b).

This work describes a fuzzy modeling methodology with the aim of getting a good interpretability–accuracy trade-off when building FRBSs for classification tasks also called fuzzy rule-based classifiers (FRBCs). The rest of the paper is structured as follows. Section 2 describes the proposed modeling process. Section 3 explains the experiments made and the obtained results. Finally, Sect. 4 draws some conclusions and future works.

2 Methodology

The starting point is the Highly Interpretable Linguistic Knowledge (HILK) fuzzy modeling methodology (Alonso et al. 2008) which focuses on making easier the design process of interpretable FRBSs. It offers an integration framework for combining both expert knowledge and knowledge extracted from data, which is likely to yield robust and compact systems.

Unfortunately, in some applications expert knowledge is missing because the expert is not able to explicitly formalize its knowledge. This situation usually turns up in complex problems involving many input variables where expert knowledge extraction and representation becomes a bottle neck for the whole modeling process. It is also common that experts only provide a partial view of the problem. They can only describe the global behavior and some specific situations. Hence, a fuzzy modeling methodology must be able to get good models even when there is no expert knowledge or it is not complete.

This work only deals with automatic learning from data taking profit of the general framework provided by HILK, including strong fuzzy partitions, global semantics, Mamdani rules, linguistic simplification, partition tuning, etc. We assume that expert knowledge is not available in the problems under consideration. Therefore, the proposed method works without including expert knowledge. In addition, it enhances HILK with some new functionalities (feature selection, interpretability-guided simplification, etc.) in order to get comprehensible FRBCs.

A FRBC is a fuzzy system able to select one class from a pre-defined set of NC classes $C = \{C^1, C^2, ..., C^{NC}\}$. Given an n-dimensional input space $(X \subseteq R^n)$, a fuzzy inference yields an activation degree associated to each class C^i. Of course, several classes can be activated at the same time with activation degree greater than zero. FRBCs designed by HILK are endowed with the usual fuzzy classification structure based on the Max–Min inference scheme, and the winner rule fuzzy reasoning mechanism:

$$y_{FRBC}(x^p) = C^i \Leftrightarrow \mu_{C^i}(x^p) = \max_{k=1,\ldots,NC} \mu_{C^k}(x^p) \tag{1}$$

$$\mu_{C^k}(x^p) = \max_{R=1,\ldots,NR} \mu_R(x^p) \Leftrightarrow Y_R \text{ is } C^k \tag{2}$$

$$\mu_R(x^p) = \min_{i=1,\ldots,NI} \mu_{A_i^j}(x_i^p) \tag{3}$$

where given an input vector $x^p = \{x_1^p, ..., x_{NI}^p\}$, the output class C^i is derived from the highest $\mu_{C^i}(x^p)$ which is the membership degree of x^p to the class C^i. It is computed as the maximum firing degree of all rules yielding C^i as output class. For each rule, the firing degree is computed as the minimum membership degree of x^p to all the attached A_i^j fuzzy set, for all the *NI* inputs.

Figure 1 shows graphically the global scheme of the proposed fuzzy modeling process. The whole process is made up of four main steps (the most relevant components will be detailed in the following sections):

1. **Feature selection** It consists in finding out the most discriminative variables. In addition, our method gives the most suitable number of labels.
2. **Partition design** The readability of fuzzy partitioning is a prerequisite to build interpretable FRBCs. It includes partition learning (automatic generation of fuzzy partitions from data) and partition selection.
3. **Rule-based learning** Linguistic rules are automatically extracted from data.
4. **Knowledge-based improvement** It is an iterative refinement process of both partitions and rules.

2.1 Feature selection

We have implemented a feature selection procedure based on the popular C4.5 algorithm introduced by Quinlan (1993) and improved in (Quinlan 1996). Figure 2 shows a simple example for a simulated problem with five inputs.

This algorithm lets us discover the most discriminative variables (V1, V3, and V5 in the example). If all inputs were used in the tree we may select only a subset of them. This would be equivalent to prune the tree at the cost of losing some information. The closer to the root, the more important a variable is. Hence, inputs are explicitly ranked according to their contribution for the information gain represented by entropy. At each node of the tree, C4.5 selects the attribute (input variable) that most effectively splits the remaining set of training samples.

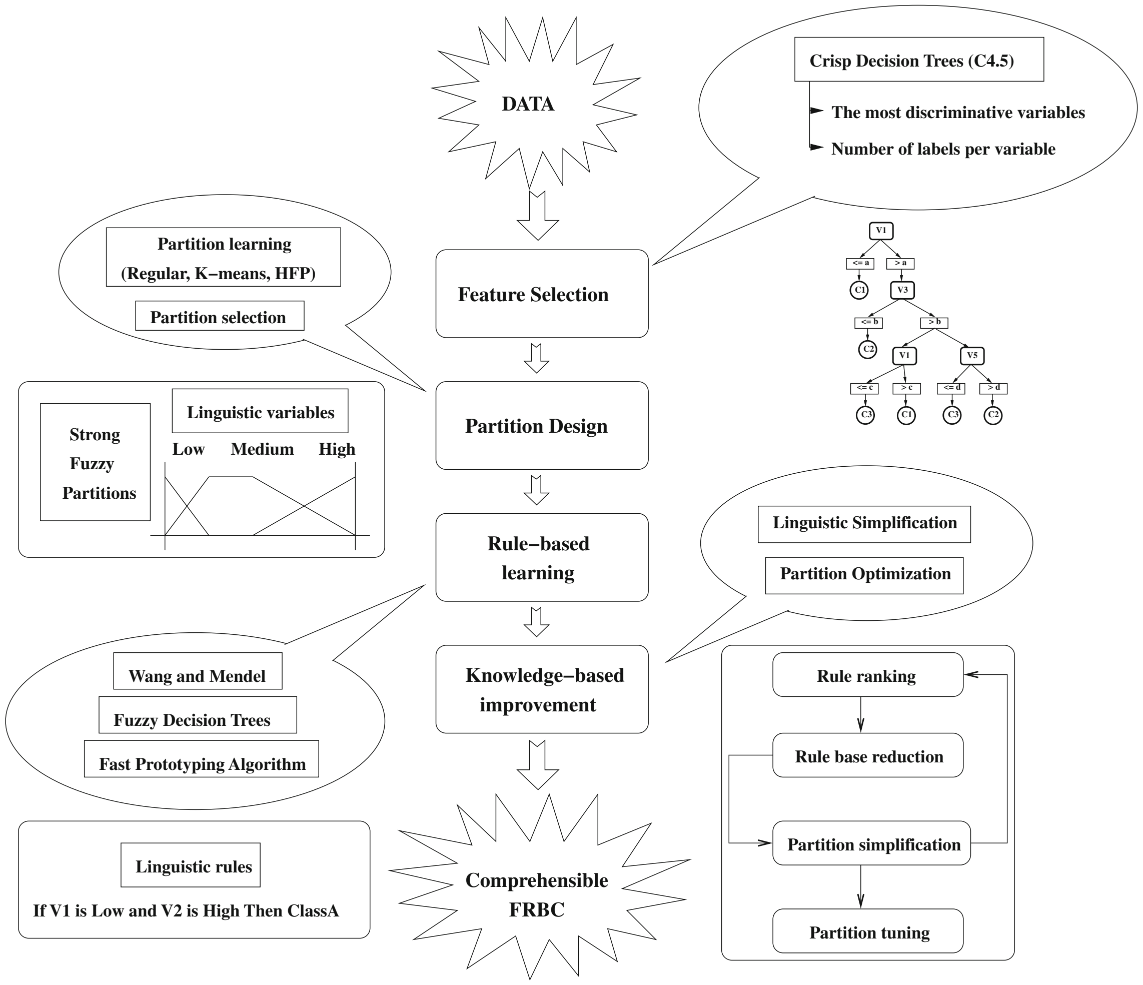

Fig. 1 Scheme of the proposed fuzzy modeling process

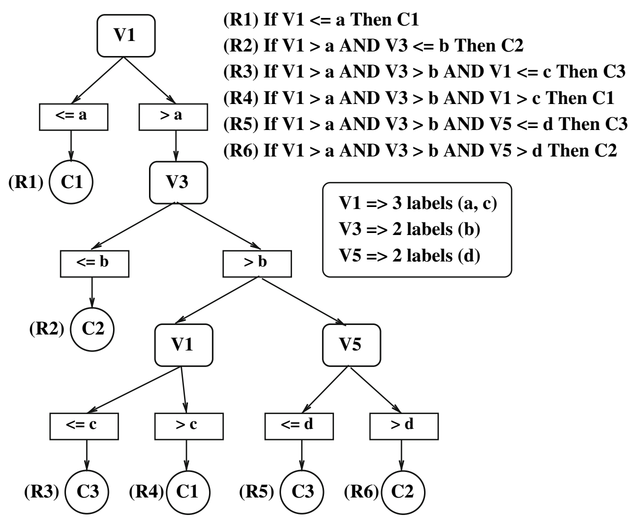
(R1) If V1 <= a Then C1
(R2) If V1 > a AND V3 <= b Then C2
(R3) If V1 > a AND V3 > b AND V1 <= c Then C3
(R4) If V1 > a AND V3 > b AND V1 > c Then C1
(R5) If V1 > a AND V3 > b AND V5 <= d Then C3
(R6) If V1 > a AND V3 > b AND V5 > d Then C2
V1 => 3 labels (a, c)
V3 => 2 labels (b)
V5 => 2 labels (d)

Fig. 2 Example of crisp decision tree

In addition, generated crisp decision trees can be easily translated into rules by reading them from the root to the leaves (Abonyi et al. 2003). Regarding our simple example, the tree plotted in Fig. 2 is translated into six rules. Observe how each branch of the tree yields a new rule.

Finally, the number of breaking values per variable appearing in a tree gives an estimation of the number of fuzzy labels needed for that variable. In our example, we need three labels for V1, while it is enough with two labels for V3 as well as for V5. Notice that, the same variable, for instance V1 in the example, can appear several times in the tree. In consequence, rule premises are expressed in terms of intervals. A linguistic proposition like (V1 $>$ a AND V1 $\leq$ c) may be rewritten (V1 is L2). Since V1 has two breaking points (a and c), the range covered by V1 [min, max] can be divided into the following intervals related to three labels:

- L1 = [min, a]
- L2 = (a, c]
- L3 = (c, max]

However, we do not care about the specific values of a and c which may bias the rest of the process. At this step, we are only interested in finding out a suitable number of terms. The tuning of parameters will be made at the end.

2.2 Partition design

Once selected the most influential variables and the number of fuzzy labels for each of them, the next step is generating the best fitted fuzzy partitions. The use of strong fuzzy partitions (SFPs) (Ruspini 1969; Loquin and Strauss 2006) has already been pointed out as an effective way of satisfying semantic constraints (normalization, coverage, distinguishability, etc.) demanded to get comprehensible partitions (De Oliveira 1999; Mencar and Fanelli 2008).

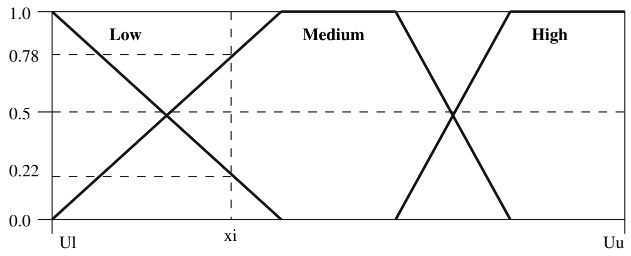

Fig. 3 A strong fuzzy partition with three fuzzy sets

Figure 3 illustrates a SFP including three fuzzy sets. As it can be seen the same value *xi* is partially *Low* (0.22) and *Medium* (0.78), but the addition of both membership degrees equals one. It is also important to highlight that psychologists (Miller 1956; Saaty and Ozdemir 2003) recommend to work with an odd number of terms (because it is easier to make reasoning around a central term) and a small (justifiable) number of terms (7 $\pm$ 2 is a limit of human information processing capability). Therefore, in our approach we never define more than nine terms for each input.

Looking for the best partition according to data distribution we propose the generation of three different partitions (with the number of labels resultant of the previous feature selection step) for each variable: (1) REG, uniformly distributed partition on the universe of discourse; (2) KM, partition defined by the centroids provided by the K-means algorithm (Hartigan and Wong 1979); and (3) HFP, partition generated by a fuzzy method guided by interpretability (Guillaume and Charnomordic 2004).

Then, generated partitions can be compared according to the three quality criteria defined by equations 4 to 6. The notation is as follows: μ_{ik} is the degree of membership of the *k-th* element of the data set to the *i-th* element of the fuzzy partition, M stands for the number of terms of the fuzzy partition, and n represents the cardinality of the data set.

$$PE = -\frac{1}{n}\left\{\sum_{k=1}^{n}\sum_{i=1}^{M}[\mu_{ik}\log_a(\mu_{ik})]\right\} \quad (4)$$

$$PC = \frac{\sum_{k=1}^{n}\sum_{i=1}^{M}\mu_{ik}^2}{n} \quad (5)$$

$$ChI = \frac{1}{n}\sum_{k=1}^{n}\max_i \mu_{ik} - \frac{2}{M(M-1)}\sum_{i=1}^{M-1}\sum_{j=i+1}^{M}\frac{1}{n}\sum_{k=1}^{n}\min(\mu_{ik},\mu_{jk}) \quad (6)$$

The partition winning at least two criteria is selected because we apply an absolute majority voting process. A good partition should minimize the partition entropy (*PE*) defined by Bezdek (1981), while maximizing both the partition coefficient (*PC*) introduced by Bezdek (1981) as well as the Chen index (*ChI*) defined by Chen (2002).

2.3 Rule-based learning

After designing all the fuzzy partitions it is time to describe the system behavior in the form of linguistic rules (Mamdani 1977):

$$R : \textbf{If}\ \underbrace{\underbrace{I_1 \text{ is } A_1^i}_{Premise\ P_1}\ \textbf{AND}\ldots\textbf{AND}\ \underbrace{I_{NI} \text{ is } A_{NI}^j}_{Premise\ P_{NI}}}_{Premise}\ \textbf{Then}\ \underbrace{Y_R \text{ is } C^i}_{Conclusion}$$

where given a rule *R*, rule premises are made up of tuples (*input variable*, *linguistic term*) where I_a is the name of the input variable *a*, while A_a^i represents the linguistic term *i* defined for such variable, with *a* belonging to {1, ..., *NI*} and being *NI* the number of inputs. In the conclusion part, C^i represents one of the possible output classes.

Note that we are imposing global semantics, i.e., all the rules use the same linguistic terms defined by the same fuzzy sets. Furthermore, we consider two kinds of linguistic terms named as elementary and composite ones. Elementary terms are those directly attached to the fuzzy sets forming the fuzzy partition. For instance, *Low*, *Medium*, and *High* for the SFP represented in Fig. 3. On the other hand, we call composite term to the convex hull of elementary terms corresponding to OR (only combinations of adjacent elementary terms are allowed) and NOT combinations. For instance, Fig. 4 shows the term *NOT(Low)* built from the partition illustrated in Fig. 3. It is equivalent to *Medium OR High* because we consider the most common *NOT* implementation, defined by the complement as $\mu_{NOT(A)}(x) = 1 - \mu_A(x)$.

Rule induction is made with three different algorithms which are able to automatically generate rules from data with the previously defined fuzzy partitions:

- **WM (Wang and Mendel 1992)** It starts by generating one rule for each data pair of the training set but new rules will compete with existing ones. As a result, WM generates complete rules (considering all the available variables) which are quite specific and likely to be simplified. They usually yield high accuracy regarding training patterns but very low accuracy with respect to test data. Its generalization ability is very poor.
- **FDT (Fuzzy Decision Tree)** It was proposed by Ichihashi et al. (1996) for generating a neuro-fuzzy decision tree from data. In addition, inputs are sorted according to their importance for minimizing the entropy as in the well-known C4.5 proposed by Quinlan. Then, the tree is translated into quite general incomplete rules because only a subset of input variables is considered.
- **FPA (Fast Prototyping Algorithm)** This rule learning method was defined by Glorennec (1999). It generates rules more general than the ones produced by WM, but at the same time more specific than the ones generated by FDT. It starts generating a grid with all possible combinations of input labels (complete rules like the ones generated by WM) and then, in an iterative process, outputs are defined removing redundancies and inconsistencies. If the number of inputs (and labels defined per input) is high then FPA is quite inefficient. Therefore it needs a previous feature selection process in order to tackle with complex problems.

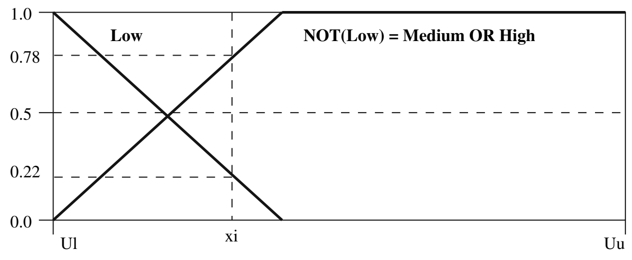

Fig. 4 An example of composite term

2.4 Linguistic simplification

With the aim of getting a more compact and general rule base HILK offers a powerful and flexible simplification procedure which affects to the whole knowledge base (KB) including both rule-based simplification and partition reduction. It starts looking for redundant elements (labels, inputs, rules, etc.) that can be removed without altering the system accuracy. Then, it tries to merge elements always used together. Finally, it forces removing elements apparently needed but not contributing too much to the final accuracy. In fact, it is an iterative process where we first act on the rules and then on the partitions at each iteration. This cycle is repeated until no more interpretability improvement is feasible without penalizing accuracy more than a predefined threshold. Depending on the complexity of the initial KB the whole simplification can take several iterations because rule-based reduction affects partition simplification and vice versa.

Let us pose a simple example. We are going to consider a classification problem with two classes (C_1 and C_2). It involves two inputs (I_1 and I_2), each of them characterized by SFPs made up of five fuzzy sets with the five related elementary terms (*Very low*, *Low*, *Medium*, *High*, *Very high*). The rule base comprises the following five rules (note that they only include elementary terms):

R^1: If I_1 is *High* **AND** I_2 is *High*

Then Y is C_1

R^2: If I_1 is *Very high* **AND** I_2 is *High*

Then Y is C_1

R^3: If I_1 is *Low* **AND** I_2 is *Medium*

Then Y is C_2

R^4: If I_1 is *Medium* **AND** I_2 is *Medium*

Then Y is C_2

R^5: If I_1 is *Medium* **AND** I_2 is *Very high*

Then Y is C_2

In the first iteration, rule-based reduction yields R^{12} which is obtained after merging R^1 and R^2, as well as R^{34} which results of merging R^3 and R^4. In consequence, the new rule base is made up of three rules (some of them include composite terms):

R^{12}: If I_1 is (*High OR Very high*) **AND** I_2 is *High*

Then Y is C_1

R^{34}: If I_1 is (*Low OR Medium*) **AND** I_2 is *Medium*

Then Y is C_2

R^5: If I_1 is *Medium* **AND** I_2 is *Very high*

Then Y is C_2

Then, partition simplification yields only four terms for I_1 (*Very low*, *Low*, *Medium*, *High OR Very high*). Note that *Low* and *Medium* can not be merged in a unique term because *Medium* is used alone in the rule R^5. Regarding I_2, it only keeps three out of the five terms initially defined (*Medium*, *High*, *Very high*).

At this point, the simplification procedure seems to be ended. However, it is time to explore if there is some element in the KB that can be deliberately removed without reducing accuracy under a predefined threshold Δ. For instance, if we may remove the premise (I_2 is *Very high*) from the rule R^5 producing a lost L_1 smaller or equal than Δ, such rule may be merged with R^{34} in the next iteration. As a result, the final rule base would include only two rules. In addition, I_2 would only have two terms (*Medium*, *High*). Notice that, removing a linguistic term implies changing the fuzzy partition (by extending the related adjacent fuzzy sets) in order to keep always a SFP. For further details, the interested reader is referred to (Alonso et al. 2008) where this simplification procedure is thoroughly explained.

Thanks to the use of global semantics rule comparison can be directly made at linguistic level. In addition, the process is absolutely deterministic. It represents a greedy search. As a result, it is human-oriented and quite intuitive. However, final results depend on the initial rule ordering. As illustrated in the previous example, some steps are allowed if and only if they do not reduce accuracy under a predefined threshold. Imagine that we start exploring R^{12} instead of R^5 with the aim of finding premises that can be deliberately removed. It may happen that if we remove the premise (I_2 is *High*) from rule R^{12}, accuracy is slightly reduced yielding a lost $L_2 \leq \Delta$, but $L_1 + L_2 > \Delta$ (where L_1 is the lost produced when beginning with R^5). In consequence, only one of these two operations can be made. If we began from R^{12} it would be simplified, but R^{34} and R^5 would not be merged later on.

One may think about exploring all possible alternatives and selecting the one yielding the highest interpretability improvement. Nevertheless, this is not easy at all because after each taken decision we have a new tree of alternatives. We need to find out the best chain of decisions taking into account that each new decision strongly depends on the previous ones. We may see it as a chess game. Moreover, in real problems we have to deal with dozens or even hundreds of rules involving many inputs. Thus, checking all alternatives is extremely time consuming and it is not feasible in practice. It is necessary to think about effective heuristics.

Therefore, in a first approach we have upgraded the simplification procedure of HILK adding a new rule ranking step previous to each simplification task. Hence, we will check only a reduced set of alternatives, those derived from the previously established ranking.

The proposed rule ranking is based on a novel interpretability index:

$$RBC = \sum_{j=1}^{NR} \left[complexity(R^j) \right] \tag{7}$$

$$complexity(R^j) = \prod_{a=1}^{NI} \left[complexity(P_a) \right] \tag{8}$$

$$complexity(P_a) = 2 - \frac{LT_a^j}{NL_a} \tag{9}$$

$$RBC = \sum_{j=1}^{NR} \left[\prod_{a=1}^{NI} \left(2 - \frac{LT_a^j}{NL_a} \right) \right] \tag{10}$$

A rule base (*RB*) is made up of a set of rules, so the total rule-based complexity (*RBC*) is given as the addition of all the r-complexity indices (*complexity*(R^j)) measured for the *NR* rules. Each rule involves a set of premises, so the complexity of a rule is measured as the product of all the p-complexity indices (*complexity*(P_a)) for the *NI* inputs used in the rule. A p-complexity index evaluates the complexity of a premise. It is computed regarding all the involved linguistic propositions P_a of form I_a *is* A_a^i that are included in the evaluated premise. Notice that, the linguistic term A_a^i which is assigned to the variable I_a can correspond to one of the NL_a elementary terms defined in the fuzzy partition of the input I_a. Of course, it can also be one of the composite terms which usually turn up as result of the merging of rules and linguistic terms made by the simplification procedure. LT_a^i counts the number of elementary terms included in A_a^i, taking the following values:

– One for elementary terms.
– Number of elementary terms combined with OR. For instance, it equals two for the expression *Low OR Medium*.
– NL_a minus one half for NOT composite terms, what penalizes NOT against OR composite terms involving all the elementary terms minus one.
– NL_a when input I_a is not considered in the rule. It means $complexity(P_a) = 1$.

This new interpretability index is based on conclusions derived from a web poll study devoted to discover the main influential aspects when assessing interpretability of fuzzy systems (Alonso et al. 2009). In short, people usually prefer rules free of NOT composite terms. That is why we penalize NOT composite terms against the equivalent OR ones. In addition, the increase of rule complexity perceived by people is not linear with the number of involved premises, so we have used product for combining complexity of premises.

2.5 Partition optimization

The last step in the whole fuzzy modeling process is devoted to increase the system accuracy while preserving the high interpretability previously achieved. It consists of a membership function tuning constrained to keep the SFP property. HILK offers two different optimization strategies (Alonso et al. 2007):

1. **SW (Solis-Wets)**: An *element-by-element* optimization procedure based on the classical local search strategy proposed by Solis and Wets (1981). It was described by Glorennec (1999) as *a hill climbing method with memorization of the previous successes*. To sum up, firstly system inputs are ranked regarding their frequency of use in the rule base. Thus, the optimization procedure starts with the inputs most frequently used. Then, the optimization is made label by label looking for the most suitable parameters. It lets increase accuracy in only a few iterations but it does not guarantee to find the global optimum. The algorithm stops when the maximum number of iterations is achieved, or the fitness function is under a predefined threshold. After modifying one label, the adjacent ones are also changed for keeping the SFP and the process comes back to the starting point.
2. **GT (Genetic-Tuning)**: An *all-in-one* optimization procedure based on a global search strategy inspired on the evolutionary processes that take place in nature. It becomes a genetic tuning process (Cordón et al. 2001). In short, a genetic algorithm (GA) (Goldberg 1989) usually starts with a population of several randomly generated solutions (chromosomes) and try to find better solutions by applying genetic operators. All system parameters are adjusted at the same time. The implementation details can be found in (Alonso et al. 2007) where GT is adapted from the proposal made by Cordón and Herrera (1997). First, the initial KB is used for building the first individual of the population. For each individual, a real-coded chromosome is generated by joining the basic parameters of all its fuzzy partition. Second, the rest of the population is randomly generated and it is made up of 60 individuals. Then, the following steps are repeated for each generation:
   – Binary tournament selection.
   – $BLX - \alpha$ crossover ($\alpha = 0.3$, probability $= 0.6$).
   – Uniform mutation (probability $= 0.1$).
   – Elitism.

Figure 5 shows an example of SFP with five fuzzy sets where we have represented all parameters handled by both strategies which consider the same coding scheme. For each fuzzy partition, the fuzzy sets centers or modal points (C_i in the figure) are adjusted through slight modifications. One parameter C_i characterizes each fuzzy set A_i, except for the trapezoidal membership functions where two parameters, C_{i1} and C_{i2}, have to be considered. The optimization procedure will move the C_i points of each partition, given as a result new $C_i^{'}$ points that define a new SFP, without any ambiguity. Notice that, both the initial number and order of linguistic terms are maintained by imposing an allowed variation interval for each parameter. This way, a very compact representation is got for the optimization procedure, while the SFP property is always kept.

As explained by Alonso et al. (2007), other ways of doing are possible. Indeed, Van Broekhoven et al. (2007) made a very similar proposal considering two parameters, C_{i1} and C_{i2}, for each fuzzy set disregarding its membership function shape. Thus, a vector of $2M$ real numbers characterizes a partition of M labels. As a result, the SFP property is kept, but not the membership function shapes. For example, a triangular function can derive to a trapezoidal one. We prefer to maintain at least the basic shape (triangular, trapezoidal, etc.), even though the slopes can

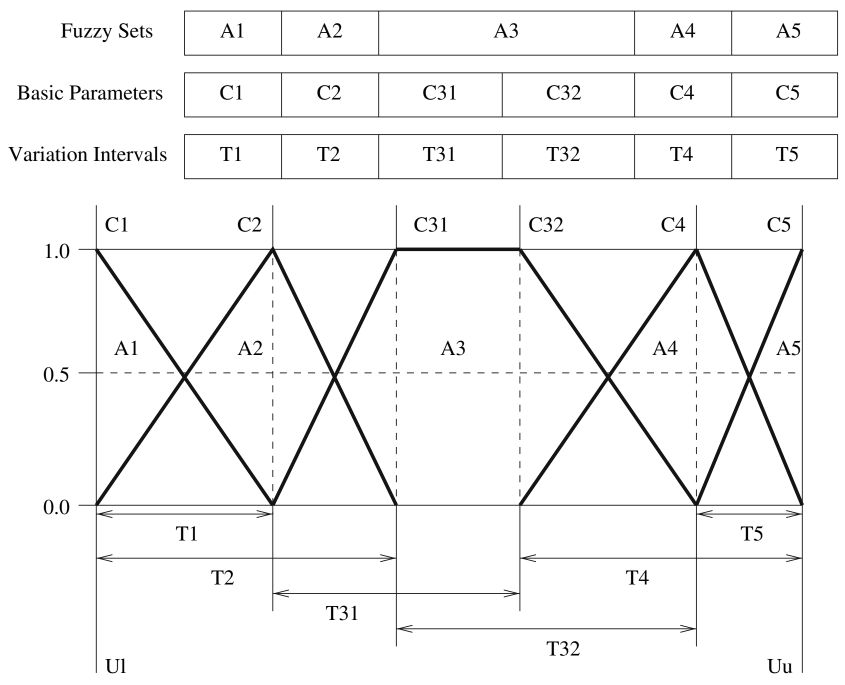

Fig. 5 Tuning parameters in a SFP

change, because it is strongly related to the linguistic term meaning. Other proposals like (Karr 1991; Cordón and Herrera 1997) coded every characteristic point of the fuzzy sets, which gives more degrees of freedom to the optimization but disregarding the SFP property. The same stands for other more recent advanced genetic tuning mechanisms (Casillas et al. 2005; Alcalá et al. 2007).

In the last years, multi-objective approaches have become very popular in the sake for fuzzy models with a good trade-off between accuracy and interpretability (Cordón et al. 2004; Ishibuchi and Nojima 2007; Herrera 2008). In addition, some authors have proposed useful indices for assessing the integrity of fuzzy partitions (Antonelli et al. 2009; Gacto et al. 2010). In such works, the search process is biased towards models that keep highly interpretable partitions in the sense that they remain close to the initial fuzzy partitions. This strategy is really useful when partitions were defined by an expert since we do not want to lose the expert knowledge in the optimization phase. However, it becomes a strong limitation when there is no expert for defining the fuzzy partitions and they are defined (in most cases) as uniformly distributed SFPs, what prevent looking for prototype values that are not represented by the uniform partition. An interpretable fuzzy partition must represent prototypes that are meaningful for the expert, but it is not necessarily uniform. For example, if we have a fuzzy variable that represents *Temperature* and it is characterized by a SFP in the range [20, 40] with three terms, a uniform partition would set three modal points corresponding to three prototype values (20, 30, 40), but everyone knows that 37° is the common temperature for human beings. Such information remained hidden if we do not have an expert and the tuning tends to keep the initial prototypes.

We have chosen single-objective optimization instead of moving towards multi-objective approaches because partition tuning only represent a small part of the whole methodology. It has to be effective and not too much demanding in terms of time and memory because at this step of the process we have already achieved a quite compact and reduced KB. Finally, the use of extended variation intervals yields the flexibility that the tuning process needs to find out hidden prototypes while keeping transparent SFPs. Of course, at the end an expert is needed for assigning the right linguistic terms to the discovered prototypes.

3 Experimental analysis

The enhanced version of HILK presented in this paper has been evaluated with six benchmark classification problems freely available from the UCI (University of California, Irvine, LA, USA) machine learning repository[1]:

- **IRIS** Database created by Fisher with the aim of classifying three varieties of the iris plant.
- **WINE** Chemical analysis of wines grown in the same region in Italy but derived from three different cultivars.

[1] http://www.ics.uci.edu/-mlearn/MLSummary.html.

Table 1 Description of the data sets under consideration

Dataset	Instances	Attributes	Classes
IRIS	150	4	3
WINE	178	13	3
GLASS	214	9	6
NEWTHYROID	215	5	3
WBCD	683	9	2
PIMA	768	8	2

- **GLASS** A study of classification of types of glass that was motivated by criminological investigation.
- **NEWTHYROID** Predicting the type of patient's thyroid disease.
- **WBCD** Classification of two cancer states (benign or malignant). Obtained from the University of Wisconsin Hospitals.
- **PIMA** Determining if a patient shows signs of diabetes according to World Health Organization criteria. All selected patients are females at least 21 years old of Pima Indian heritage.

For all the six problems listed above the comprehensibility of the classifier is highly appreciated. The first three data sets are well-known general purpose classification problems while the three remaining ones are related to medical applications. Their main characteristics are summarized in Table 1.

We have chosen 10-fold cross-validation as evaluation methodology. Cross-validation is a method for estimating generalization error based on resampling (Hjorth 1994; Plutowski et al. 1994). The same process is repeated for all the six problems previously introduced. The data set is divided into ten parts of equal size keeping the original distribution (percentage of elements for each class) in the whole set. One part is used as test set whereas the remainders are used as training set. Notice that, all tested algorithms are implemented in Fispro[2] and/or KBCT[3], two free software tools for designing FRBSs.

From now on we describe the experimental results obtained for the four stages that the proposed methodology comprises. For the sake of clarity, results are grouped and explained in the two following subsections.

3.1 Feature selection and partition design

Let us begin with the analysis of the two first modeling stages. First, the feature selection procedure identifies the most relevant inputs, yielding also their most suitable number of labels. Second, three partitions (REG, KM, and HFP) are generated for each input, and then compared to select the best one according to data distribution.

Table 2 exposes the average results (over 10-fold cross-validation) which are presented in terms of the Mean and the Standard Deviation (SD). The first part of the table (NOI and NOL) is related to the feature selection process. NOI stands for the number of selected inputs while NOL is the number of linguistic terms defined per input. The Mean for NOL is always kept under 4. As a result, we build interpretable partitions implemented as SFPs with a small number of terms. The second part of the table (REG, KM, and HFP) focuses on the partition design phase. Each column is related to one kind of fuzzy partition. Thus, looking carefully at the six columns (Mean and SD) in the center of the table it is easy to appreciate how many inputs (in average) use each kind of partition. For instance, the 8.7 inputs for GLASS are interpreted as follows: 0.9 inputs have attached uniform partitions (REG); 5.7 inputs use partitions induced with K-means (KM); and 2.1 inputs correspond to partitions generated by means of a hierarchical method (HFP). The last part of the table reports the runtime which is measured in seconds. As expected, the runtime grows with the number of attributes and instances included in the data set. Anyway, it is kept reasonably small for all the six analyzed problems.

From Table 2 we can deduce that most of the selected partitions are KM. Therefore, if we are working in a problem where one of the main requirements is that runtime must be as small as possible we can save a few seconds by only considering KM during the partition design stage. As expected, results show that automatically generated partitions are the best fitted for the data distribution. Although many authors usually advocate working directly with REG partitions by claiming they are the most interpretable ones, in lots of problems the use of non-uniform partitions can be quite useful, especially when looking for hidden prototypes. Nevertheless, it is important to remark that we are using highly interpretable partitions even when they are not always uniform.

Figure 6 shows an example of fuzzy partitions for the WINE problem. On the left side of Fig. 6a, we have plotted the partitions obtained after feature selection and partition design. Only three out of the thirteen attributes of WINE are considered. In addition, the number of terms is quite small. In consequence, you can appreciate how much interpretable the designed partitions are.

3.2 Rule-based learning and knowledge-based improvement

Once all partitions have been designed it is time to tackle with the generation of the rule base, and the subsequent

[2] http://www.inra.fr/internet/Departements/MIA/M/fispro/.

[3] http://www.mat.upm.es/projects/advocate/kbct.htm.

Table 2 Results of the experimentation (feature selection and partition design)

Dataset	NOI		NOL		REG		KM		HFP		Runtime (s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
IRIS	2	0	2.85	0.24	0	0	2	0	0	0	2	1.49
WINE	3.9	0.87	2.15	0.16	0	0	3.9	0.87	0	0	13.4	4.19
GLASS	8.7	0.48	3.63	0.31	0.9	0.87	5.7	1.88	2.1	1.2	8.7	1.83
NEWTHYROID	4.4	0.7	2.6	0.13	0	0	3.1	0.74	1.3	0.67	8.3	1.89
WBCD	5.9	1.2	2.46	0.28	0	0	5.8	1.4	0.1	0.3	3.5	4.19
PIMA	7.2	0.92	3.23	0.76	0.7	0.67	5.4	0.97	1.1	1.1	27.7	3.83

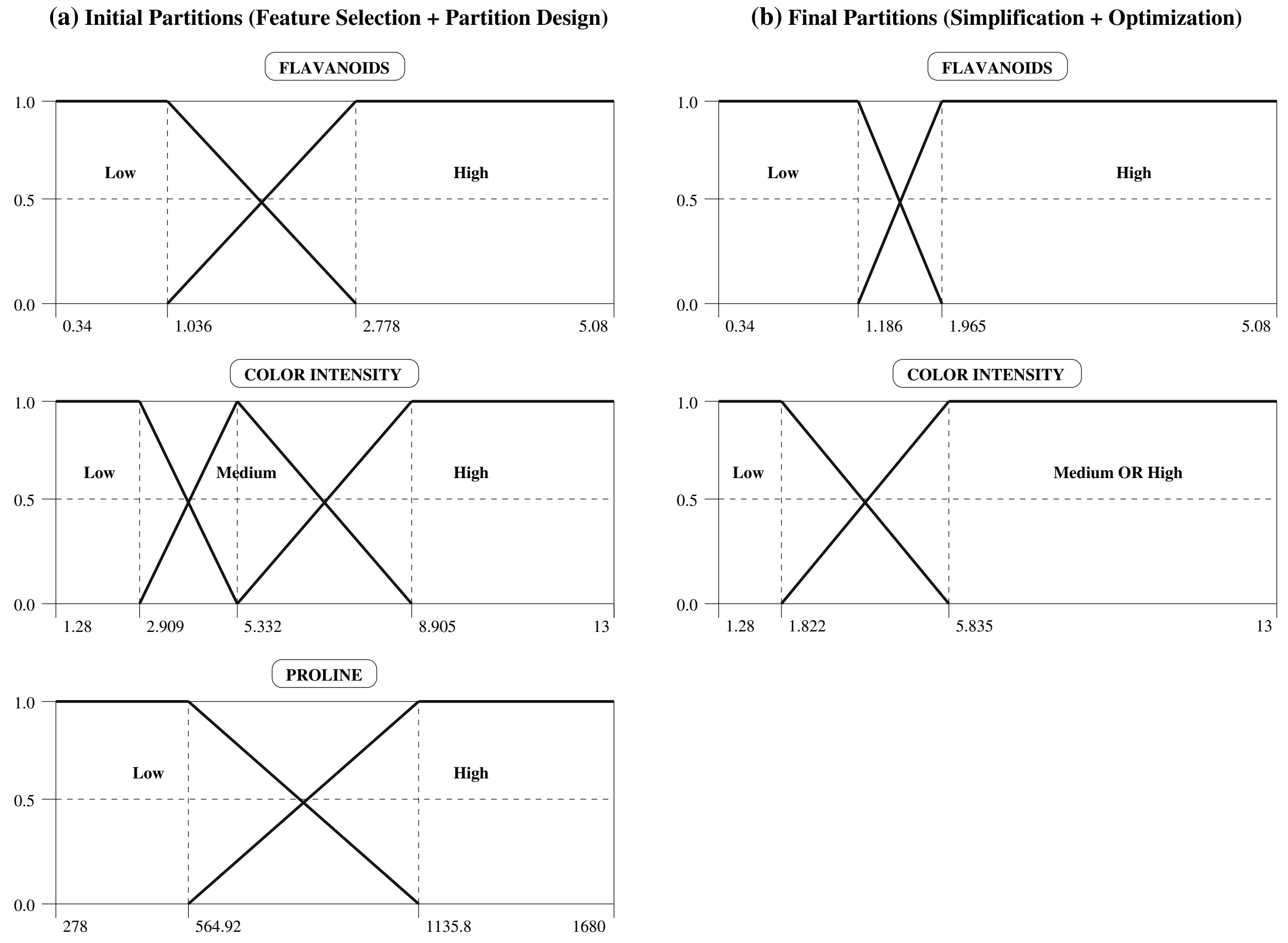

Fig. 6 Example of fuzzy partitions at the beginning (**a**) and at the end (**b**) of the modeling process for one of the ten folds in the WINE problem

improvement of both partitions and rules, as it will be explained later. On the right side of Fig. 6b, we have illustrated the same partitions plotted on the left (a) but at the end of the whole modeling process. After simplification only two inputs are kept because *Proline* has been removed. Furthermore, *Color intensity* has only two linguistic terms because *Medium* and *High* has been merged as *Medium OR High*. The effect of the optimization stage is a modification of the fuzzy partitions. They keep their original shape (triangular, trapezoidal, etc.) but their modal points (fuzzy set centers) are slightly moved. This means slopes may be altered. However, let us highlight that we only show the final partitions, which are highly interpretable, to the final user of the system. Of course, we should ask an expert (or directly to the final user) for confirming the suitability of the attached linguistic terms. The designer should wonder if such terms are really meaningful for the final user.

Coming back to the experimental analysis, after feature selection and partition design, the rest of the experimental procedure was as follows. First, rules were automatically derived from data with the three rule induction algorithms (WM, FDT, and FPA) introduced in Sect. 2.3. Then, each KB was simplified four times (with threshold $\Delta = 0.1$) exploring four different rule ranking options:

- **S** Simplification without changing the rule ranking provided by the rule induction algorithm.
- **S-IC** Rule ranking from the simplest rule to the most complex one (IC stands for increasing complexity).
- **S-DC** It is just the inverse ranking (decreasing complexity), from the most complex rule to the simplest one.
- **S-IC-DC** The three previous strategies are run in parallel. At each intermediate simplification step the solution yielding the smallest complexity is selected as the KB to be simplified in the next step.

Afterwards, the simplified KBs were compared and only the simplest one, according to *RBC* defined by Eq. 10, was selected (set in boldface in Tables 3, 4, 5, 6, 7, 8 and emphasized with symbol [*]) to be optimized. Notice that, we are evaluating the complexity of the rules regarding the complexity of all involved linguistic propositions. Hence, for each problem and for each rule induction algorithm, only one KB is selected to be optimized. We choose the simplest KB, i.e., the one yielding the smallest RBC, but not necessarily the most compact one in terms of the rest of interpretability indicators. Moreover, we have highlighted with the symbol [+] the solution yielding the smallest RBC in the tables, choosing the most accurate solution, first looking at ACC (training) and then looking at ACC (test) if needed, in the case of a draw.

Thus, Tables 3, 4, 5, 6, 7, 8 present the achieved results for the two last modeling stages (*Rule-based learning* and *Knowledge-based improvement*). In addition to rule induction and simplification, tables include results after partition optimization by the two strategies described in Sect. 2.5. Solis-Wets (O-SW) was carried out label by label with maximum number of iterations equals 10. The maximum number of generations for Genetic-Tuning (O-GT) was set to 600. The rest of parameters were defined in Sect. 2.5. Each strategy was run three times for each fold yielding a total of 30 runs for each problem. Furthermore, for comparison purpose, the first two rows in the tables show results provided by other methods implemented in Weka[4]: MP (Multilayer Perceptron) which yields very accurate neural network classifiers (disregarding interpretability), and C45 (Quinlan's decision trees) which are recognized because they provide a good trade-off between interpretability and accuracy, at the cost of achieving usually accuracy smaller than MP.

Notice that, with the aim of making a fair comparison with the previous version of HILK, we have included in all the tables three rows (PD-FDT, S, and O-SW) just below C45 and MP. They report results provided by HILK without feature selection (FS). First, we generated fuzzy partitions made up of five terms for all the attributes. The partition design (PD) includes the election of the best partition (comparing REG, KM, and HFP) for each attribute. Then, rule induction was made with FDT. We chose FDT because in our previous work (Alonso et al. 2008) it seemed to outperform the other induction methods in terms of both accuracy and interpretability. Of course, results presented in the current work are not directly comparable with those reported by Alonso et al. (2008). There, we also dealt with the combination of expert and induced knowledge, but remind the current paper is only considering induced knowledge. Then, the simplification procedure (S) was performed but without considering the ranking choice proposed in this work. Finally, the optimization stage was run with O-SW strategy which was the only one included in the first version published of HILK (Alonso et al. 2008).

Average results (over 10-fold cross-validation) are presented in terms of accuracy, interpretability, and runtime. Accuracy (ACC) is computed as the ratio of samples correctly classified for training and test. Interpretability is characterized by the total number of labels defined for all the inputs (NOI*NOL); the number of rules (NR); the total rule length (TRL) computed as the total number of premises for all the rules; the average rule length (ARL) defined as the average number of premises per rule; and the rule-base complexity (RBC) computed by equation 10 as explained in Sect. 2.4. Runtime is measured in seconds. For each indicator, we have included the Mean and the Standard Deviation (SD) over the ten folds (the 30 runs in the case of the optimization stage). Notice that, we have observed that sometimes the values computed for SD are quite large due to the diversity among the different folds.

Let us start discussing the results stored in Table 3 which corresponds to the two first problems (IRIS and WINE). They are two very well-known benchmark classification problems, probably the two most thoroughly considered in the specialized literature.

In the case of IRIS, the problem is so simple that our method is able to achieve almost the ideal solution (only one rule per class, only one premise per rule, and accuracy close to 100% regarding both training and test patterns). The best results (look at Table 3) were provided by FS-PD-FDT-S-O-GT, i.e., rules were first induced with FDT after feature selection (FS) and partition design (PD), then they

[4] http://www.cs.waikato.ac.nz/ml/weka/.

Table 3 Results of the experimentation (rule-based learning, linguistic simplification, and partition optimization) (IRIS)

	ACC (training)		ACC (test)		NOI*NOL		NR		TRL		ARL		RBC		Runtime (s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
MP	**0.9874**	**0.003**	**0.973**	**0.034**	–	–	–	–	–	–	–	–	–	–	–	–
C45	**0.98**	**0.003**	**0.96**	**0.056**	–	–	**4.7**	**0.483**	**12.5**	**2.415**	**2.635**	**0.266**	–	–	–	–
IRIS																
HILK																
PD-FDT	0.9793	0.007	0.9532	0.045	20	0	16.4	5.5	39.9	17.72	2.341	0.404	81.147	38.678	0.7	0.258
S	0.9801	0.006	0.9466	0.052	10.7	2.16	6.5	1.58	11.7	4.83	1.745	0.299	18.358	7.96	7.1	0.994
O-SW	0.983	0.005	0.9466	0.052	10.7	2.16	6.5	1.58	11.7	4.83	1.745	0.299	18.358	7.96	1.4	0.516
HILK++																
FS-PD-WM	0.8376	0.093	0.8202	0.104	5.7	0.48	4.6	0.516	9.2	1.033	2	0	11.925	1.556	0.55	0.16
S [*]	**0.8376**	**0.093**	**0.8202**	**0.104**	**3.2**	**0.42**	**3**	**0**	**3.6**	**1.265**	**1.2**	**0.422**	**5.35**	**0.738**	**2.2**	**0.42**
S-IC	0.8376	0.093	0.8202	0.104	3.2	0.42	3	0	3.6	1.265	1.2	0.422	5.35	0.738	5.1	0.57
S-DC	0.8376	0.093	0.8202	0.104	3.2	0.42	3	0	3.6	1.265	1.2	0.422	5.35	0.738	4.3	0.67
S-IC-DC	0.8376	0.093	0.8202	0.104	3.2	0.42	3	0	3.6	1.265	1.2	0.422	5.35	0.738	3.1	0.57
O-SW	**0.9595**	**0.005**	**0.94**	**0.058**	**3.2**	**0.42**	**3**	**0**	**3.6**	**1.265**	**1.2**	**0.422**	**5.35**	**0.738**	**0.95**	**0.095**
O-GT	0.9595	0.005	0.935	0.068	3.2	0.42	3	0	3.6	1.265	1.2	0.422	5.35	0.738	339.47	12.11
FS-PD-FDT	0.9073	0.03	0.9001	0.065	5.7	0.48	3.7	0.483	3.7	0.483	1	0	6.4	0.966	0.6	0.21
S [*]	**0.9073**	**0.03**	**0.9001**	**0.065**	**3**	**0**	**3**	**0**	**3**	**0**	**1**	**0**	**5**	**0**	**2.3**	**1.16**
S-IC	0.9073	0.03	0.9001	0.065	3	0	3	0	3	0	1	0	5	0	3.3	0.95
S-DC	0.9073	0.03	0.9001	0.065	3	0	3	0	3	0	1	0	5	0	2.9	0.74
S-IC-DC	0.9073	0.03	0.9001	0.065	3	0	3	0	3	0	1	0	5	0	2.1	0.87
O-SW	0.9609	0.006	0.94	0.058	3	0	3	0	3	0	1	0	5	0	0.917	0.19
O-GT [+]	**0.9609**	**0.006**	**0.9511**	**0.056**	**3**	**0**	**3**	**0**	**3**	**0**	**1**	**0**	**5**	**0**	**329.73**	**27.42**
FS-PD-FPA	0.9073	0.03	0.9001	0.065	5.7	0.48	5	0	10	0	2	0	12.9375	0.302	0.6	0.21
S [*]	**0.9073**	**0.03**	**0.9001**	**0.065**	**3**	**0**	**3**	**0**	**3**	**0**	**1**	**0**	**5**	**0**	**2.2**	**0.42**
S-IC	0.9073	0.03	0.9001	0.065	3	0	3	0	3	0	1	0	5	0	4.8	0.63
S-DC	0.9073	0.03	0.9001	0.065	3	0	3	0	3	0	1	0	5	0	4	0
S-IC-DC	0.9073	0.03	0.9001	0.065	3	0	3	0	3	0	1	0	5	0	3.3	0.48
O-SW	0.9609	0.006	0.94	0.058	3	0	3	0	3	0	1	0	5	0	0.917	0.18
O-GT	**0.9609**	**0.006**	**0.944**	**0.054**	**3**	**0**	**3**	**0**	**3**	**0**	**1**	**0**	**5**	**0**	**328.23**	**18.61**

Table 4 Results of the experimentation (rule-based learning, linguistic simplification, and partition optimization) (WINE)

	ACC (training)		ACC (test)		NOI*NOL		NR		TRL		ARL		RBC		Runtime (s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
MP	**1**	**0**	**0.9719**	**0.039**	–	–	–	–	–	–	–	–	–	–	–	–
C45	**0.9881**	**0.006**	**0.9385**	**0.055**	–	–	**5.4**	**0.699**	**13.7**	**3.057**	**2.513**	**0.204**	–	–	–	–
WINE																
HILK																
PD-FDT	1	0	0.9384	0.055	65	0	46.3	3.59	127	16.98	2.73	0.17	284906.3	18533.9	1.2	0.42
S	1	0	0.9214	0.059	23	2.36	12.3	1.25	31.1	3.38	2.53	0.11	1782.94	1279.07	28.2	4.05
O-SW	1	0	0.9214	0.059	23	2.36	12.3	1.25	31.1	3.38	2.53	0.11	1782.94	1279.07	3.2	0.63
HILK++																
FS-PD-WM	0.8714	0.027	0.8712	0.105	8.3	1.49	13.1	5.04	54.2	32.29	3.9	0.87	77.74	57.87	0.65	0.24
S	0.8738	0.024	0.8712	0.105	8.1	1.73	5.1	1.59	15	8.77	2.77	0.63	18.18	12.97	5.3	1.42
S-IC [*]	**0.8726**	**0.026**	**0.8712**	**0.105**	**7.9**	**1.59**	**5.1**	**1.59**	**14.4**	**8.18**	**2.67**	**0.55**	**16.11**	**9.46**	**5.5**	**1.84**
S-DC	0.8738	0.025	0.8712	0.105	8.1	1.73	5.1	1.59	15.2	8.73	2.82	0.64	18.41	12.93	5.4	1.78
S-IC-DC	0.8738	0.025	0.8712	0.105	8.1	1.73	5.1	1.59	14.9	8.49	2.76	0.6	17.9	12.18	9.3	3.2
O-SW	0.9605	0.016	**0.9103**	**0.08**	7.9	1.59	5.1	1.59	14.4	8.18	2.67	0.55	16.11	9.46	5.53	2
O-GT	**0.9723**	**0.01**	0.9029	0.06	**7.9**	**1.59**	**5.1**	**1.59**	**14.4**	**8.18**	**2.67**	**0.55**	**16.11**	**9.46**	**639.53**	**78.35**
FS-PD-FDT	0.8788	0.028	0.8546	0.105	8.3	1.49	5.1	0.99	11.7	4.92	2.22	0.47	14.18	6.71	0.6	0.21
S	0.8788	0.028	0.8546	0.105	6.4	1.26	4.4	0.97	9.7	3.71	2.14	0.33	11.06	4.34	4.5	2.17
S-IC	0.8788	0.028	0.8546	0.105	6.4	1.26	4.4	0.97	9.7	3.71	2.14	0.33	11.06	4.34	3.3	1.25
S-DC [*]	**0.8807**	**0.028**	**0.849**	**0.105**	**6.6**	**1.64**	**4.3**	**0.82**	**9.5**	**3.47**	**2.15**	**0.35**	**10.89**	**4.19**	**3.2**	**1.32**
S-IC-DC	0.8788	0.028	0.8546	0.105	6.4	1.26	4.4	0.97	9.7	3.71	2.14	0.33	11.06	4.34	5.3	2.06
O-SW	0.96	0.016	0.9101	0.048	6.6	1.64	4.3	0.82	9.5	3.47	2.15	0.35	10.89	4.19	3.3	1.01
O-GT [+]	**0.9626**	**0.016**	**0.9119**	**0.046**	**6.6**	**1.64**	**4.3**	**0.82**	**9.5**	**3.47**	**2.15**	**0.35**	**10.89**	**4.19**	**579.86**	**20.58**
FS-PD-FPA	0.8896	0.025	0.8825	0.076	8.3	1.49	15.7	6.13	65.6	39.74	3.9	0.87	94.29	70.02	0.65	0.24
S	0.8908	0.026	0.8714	0.083	7.5	1.84	4.7	0.95	12.9	5.24	2.66	0.52	15.22	7.16	5.8	1.32
S-IC [*]	**0.892**	**0.025**	**0.8825**	**0.084**	**7.7**	**1.25**	**4.7**	**0.67**	**12.3**	**3.37**	**2.58**	**0.35**	**13.46**	**3.95**	**5.3**	**1.77**
S-DC	0.8908	0.026	0.8714	0.083	7.5	1.84	4.7	0.95	12.9	5.24	2.66	0.52	15.22	7.16	5.1	1.45
S-IC-DC	0.8908	0.026	0.8714	0.083	7.5	1.84	4.7	0.95	12.5	5.04	2.58	0.45	14.6	6.9	9.1	3.18
O-SW	0.9482	0.017	**0.9043**	**0.059**	7.7	1.25	4.7	0.67	12.3	3.37	2.58	0.35	13.46	3.95	3.73	1.99
O-GT	**0.9564**	**0.017**	0.8899	0.058	**7.7**	**1.25**	**4.7**	**0.67**	**12.3**	**3.37**	**2.58**	**0.35**	**13.46**	**3.95**	**626.76**	**83.73**

Table 5 Results of the experimentation (rule-based learning, linguistic simplification, and partition optimization) (GLASS)

	ACC (training)		ACC (test)		NOI*NOL		NR		TRL		ARL		RBC		Runtime (s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
MP	**0.8177**	**0.025**	**0.69**	**0.068**	–	–	–	–	–	–	–	–	–	–	–	–
C45	**0.9257**	**0.018**	**0.6586**	**0.089**	–	–	**24**	**1.89**	**142.7**	**14.65**	**5.94**	**0.29**	–	–	–	–
GLASS																
HILK																
PD-FDT	0.863	0.036	0.654	0.074	45	0	116.8	22.12	467.9	112.73	3.97	0.22	1811.52	770.08	5.4	0.7
S	0.8609	0.036	0.6543	0.08	34.3	2.98	40.3	5.87	147.4	27.05	3.64	0.14	407.44	94.52	69.6	17.14
O-SW	0.8718	0.031	0.6636	0.079	34.3	2.98	40.3	5.87	147.4	27.05	3.64	0.14	407.44	94.52	16.6	5.83
HILK++																
FS-PD-WM	0.8034	0.049	0.6071	0.11	31.5	1.78	90.1	14.78	783.9	132.5	8.7	0.483	8203.6	2131.05	0.75	0.26
S	0.7881	0.049	0.6117	0.112	30.1	2.38	34.1	6.38	275.6	54.72	8.07	0.39	1961.72	597.29	39	9.21
S-IC [*]	**0.7893**	**0.05**	**0.6118**	**0.107**	**30.6**	**2.06**	**35.3**	**7.18**	**254.9**	**66.22**	**7.18**	**0.897**	**1576.09**	**760.72**	**50.7**	**19.11**
S-DC	0.7887	0.05	0.5931	0.107	30	2.98	34.4	6.72	280.8	67.53	8.098	0.67	2110.48	869.1	52.7	13.55
S-IC-DC	0.7893	0.05	0.5926	0.099	30.4	2.27	34.5	5.99	278.8	54.22	8.06	0.49	2067.41	677.22	139.9	41.17
O-SW	0.7928	0.046	0.612	0.094	30.6	2.06	35.3	7.18	254.9	66.22	7.18	0.897	1576.09	760.72	26.6	4.62
O-GT	**0.82**	**0.028**	**0.66**	**0.093**	**30.6**	**2.06**	**35.3**	**7.18**	**254.9**	**66.22**	**7.18**	**0.897**	**1576.09**	**760.72**	**1386**	**141.1**
FS-PD-FDT	0.8074	0.033	0.6452	0.076	31.5	1.78	72	21.14	271	95.6	3.7	0.32	728.64	324.29	2.6	0.52
S	0.7923	0.033	0.6547	0.09	28.9	3.9	31.4	5.54	109.7	23.76	3.47	0.27	248.06	70.77	27.9	9.88
S-IC	0.7898	0.034	0.6404	0.096	29	4	32.7	6.53	114.9	29.65	3.48	0.29	261.4	99.6	37.9	11.44
S-DC [*]	**0.7903**	**0.034**	**0.6547**	**0.09**	**29**	**3.59**	**31.6**	**6.19**	**108.9**	**28.25**	**3.41**	**0.28**	**244.92**	**94.04**	**43**	**13.61**
S-IC-DC	0.7898	0.034	64.04	0.096	29.2	3.67	32.6	6.2	114.9	27.93	3.49	0.27	266.05	94.85	95.9	31.01
O-SW	0.8077	0.039	**0.6735**	**0.082**	29	3.59	31.6	6.19	108.9	28.25	3.41	0.28	244.92	94.04	17.6	7.17
O-GT [+]	**0.8252**	**0.033**	0.661	0.095	**29**	**3.59**	**31.6**	**6.19**	**108.9**	**28.25**	**3.41**	**0.28**	**244.92**	**94.04**	**521.3**	**69.63**
FS-PD-FPA	0.6387	0.026	0.5613	0.079	31.5	1.78	67.2	11.07	587	114.42	8.7	0.483	6190.1	1903.9	2.4	0.84
S	0.6225	0.03	0.5565	0.08	27.6	1.84	15.7	2.5	127.7	26.22	8.1	0.64	758.87	248.86	31.4	9.96
S-IC [*]	**0.6277**	**0.025**	**0.5663**	**0.066**	**27.6**	**1.71**	**14.8**	**3.55**	**106.3**	**24.63**	**7.22**	**0.72**	**458.28**	**125.9**	**45.2**	**13.09**
S-DC	0.6235	0.03	0.5657	0.07	27.6	1.84	15.8	2.7	128.1	26.47	8.087	0.6	756.52	232.84	42.3	11.9
S-IC-DC	0.6251	0.03	0.58	0.068	27.7	1.57	15.6	3.06	120.4	33.37	7.64	0.8	652.07	281.82	99.1	34.5
O-SW	0.6444	0.028	**0.5618**	**0.087**	27.6	1.71	14.8	3.55	106.3	24.63	7.22	0.72	458.28	125.9	24.5	7.47
O-GT	**0.6969**	**0.02**	**0.5562**	**0.083**	**27.6**	**1.71**	**14.8**	**3.55**	**106.3**	**24.63**	**7.22**	**0.72**	**458.28**	**125.9**	**587.43**	**59.51**

Table 6 Results of the experimentation (rule-based learning, linguistic simplification, and partition optimization) (NEWTHYROID)

	ACC (training)		ACC (test)		NOI*NOL		NR		TRL		ARL		RBC		Runtime (s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
MP	**0.9886**	**0.004**	**0.9675**	**0.031**	–	–	–	–	–	–	–	–	–	–	–	–
C45	**0.9829**	**0.0043**	**0.9205**	**0.063**	–	–	**8**	**0.943**	**31.3**	**6.395**	**3.874**	**0.406**	–	–	–	–
NEWTHYROID																
HILK																
PD-FDT	0.9504	0.01	0.9256	0.058	25	0	32.1	7.17	87	25	2.68	0.2	188.26	68.17	0.55	0.16
S	0.9504	0.01	0.9212	0.068	16.5	2.12	10.4	1.77	25.1	4.68	2.41	0.16	37.44	7.96	11.9	2.73
O-SW	0.9694	0.013	0.9443	0.06	16.5	2.12	10.4	1.77	25.1	4.68	2.41	0.16	37.44	7.96	5.2	2.09
HILK++																
FS-PD-WM	0.8668	0.023	0.8375	0.081	11.4	1.58	14.1	3.48	63.5	20.87	4.4	0.699	114	45.67	0.5	0
S	0.8698	0.023	0.8468	0.086	9.2	2.04	5.3	1.25	16.1	6.54	2.95	0.62	20.92	10.13	4.5	1.84
S-IC [*]	**0.8756**	**0.03**	**0.8424**	**0.089**	**9**	**2.3**	**5.5**	**1.51**	**15.9**	**7.68**	**2.78**	**0.708**	**20.51**	**11.19**	**4.4**	**1.17**
S-DC	0.8698	0.023	0.8468	0.086	9.2	2.04	5.3	1.25	16	6.98	2.91	0.69	20.97	10.61	10.5	2.76
S-IC-DC	0.8735	0.027	0.847	0.089	9.3	2.26	5.5	1.18	16.4	6.26	2.899	0.62	21.48	8.93	7.8	2.09
O-SW	0.9318	0.025	0.8747	0.064	9	2.3	5.5	1.51	15.9	7.68	2.78	0.708	20.51	11.19	3.067	1.6
O-GT	**0.9631**	**0.014**	**0.898**	**0.07**	**9**	**2.3**	**5.5**	**1.51**	**15.9**	**7.68**	**2.78**	**0.708**	**20.51**	**11.19**	**317.1**	**23.46**
FS-PD-FDT	0.8733	0.026	0.856	0.041	11.4	1.58	7.4	1.43	14.1	3.69	1.883	0.188	20.27	5.54	0.65	0.24
S [*]	**0.8738**	**0.027**	**0.856**	**0.041**	**8.1**	**1.91**	**6**	**1.33**	**11.1**	**3.69**	**1.815**	**0.217**	**15.26**	**5.28**	**2.7**	**0.67**
S-IC	0.8738	0.027	0.856	0.041	8.1	1.91	6	1.33	11.1	3.69	1.815	0.217	15.26	5.28	2.6	0.52
S-DC	0.8738	0.027	0.856	0.041	8.1	1.91	6	1.33	11.1	3.69	1.815	0.217	15.26	5.28	2.6	0.53
S-IC-DC	0.8738	0.027	0.856	0.041	8.1	1.91	6	1.33	11.1	3.69	1.815	0.217	15.26	5.28	4.3	0.82
O-SW	0.9386	0.018	0.907	0.069	8.1	1.91	6	1.33	11.1	3.69	1.815	0.217	15.26	5.28	2.9	1.94
O-GT	**0.9567**	**0.01**	**0.9208**	**0.077**	**8.1**	**1.91**	**6**	**1.33**	**11.1**	**3.69**	**1.815**	**0.217**	**15.26**	**5.28**	**258.2**	**29.81**
FS-PD-FPA	0.874	0.026	0.8696	0.044	11.4	1.58	13.8	3.36	62.2	20.5	4.4	0.699	111.76	45.11	0.65	0.24
S	0.8797	0.025	0.8701	0.036	9	1.49	4.6	0.966	14.4	4.9	3.063	0.747	17.42	5.93	3.8	0.79
S-IC	0.8755	0.026	0.8696	0.044	9.4	1.78	5.1	0.99	15.7	5.44	3.024	0.74	19.86	8.75	4.6	1.65
S-DC	0.8797	0.025	0.8701	0.036	8.9	1.45	4.6	0.966	13.8	4.44	2.945	0.693	16.79	5.42	23.6	9.82
S-IC-DC [*]	**0.8808**	**0.025**	**0.8701**	**0.036**	**8.6**	**1.5**	**4.6**	**0.966**	**12**	**4.67**	**2.53**	**0.656**	**13.96**	**5.63**	**7.8**	**1.87**
O-SW	0.9303	0.011	0.8982	0.06	8.6	1.5	4.6	0.966	12	4.67	2.53	0.656	13.96	5.63	2.52	1.06
O-GT [+]	**0.9535**	**0.014**	**0.9208**	**0.053**	**8.6**	**1.5**	**4.6**	**0.966**	**12**	**4.67**	**2.53**	**0.656**	**13.96**	**5.63**	**226.76**	**35.69**

Table 7 Results of the experimentation (rule-based learning, linguistic simplification, and partition optimization) (WBCD)

	ACC (training)		ACC (test)		NOI*NOL		NR		TRL		ARL		RBC		Runtime (s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
MP	**0.992**	**0.002**	**0.9604**	**0.029**	–	–	–	–	–	–	–	–	–	–	–	–
C45	**0.9801**	**0.003**	**0.9604**	**0.024**	–	–	**10.5**	**2.37**	**41.9**	**14.25**	**3.89**	**0.59**	–	–	–	–
WBCD																
HILK																
PD-FDT	1	0	0.9634	0.022	45	0	97.8	7.13	356	33	3.63	0.11	781.19	137.69	4.5	0.53
S	1	0	0.9591	0.03	33.4	2.27	29.9	3.03	99.9	11.55	3.34	0.12	227.76	41.22	61.6	11.13
O-SW	1	0	0.9591	0.03	33.4	2.27	29.9	3.03	99.9	11.55	3.34	0.12	227.76	41.22	16.4	2.84
HILK++																
FS-PD-WM	0.9662	0.012	0.9518	0.025	14.7	3.13	92.3	39.36	589	311.94	6	1.15	1779.83	1310.55	0.7	0.26
S	0.9681	0.012	0.9531	0.027	14.4	3.34	12.8	6.18	60.7	41.39	4.3	1.16	128.05	115.32	35	20.76
S-IC [*]	**0.9673**	**0.012**	**0.959**	**0.024**	**14.4**	**3.34**	**12**	**5.05**	**52.4**	**32.9**	**3.99**	**1.06**	**80.71**	**68.54**	**45.5**	**25.4**
S-DC	0.9676	0.011	0.956	0.028	14.4	3.06	11.5	5.42	51.8	36.95	4.07	1.13	103.04	100.52	28.5	16.59
S-IC-DC	0.9677	0.011	0.952	0.027	14.2	3.15	12.7	5.12	56	33.69	4.08	1.01	99.63	83.35	69.9	49.53
O-SW	0.9771	0.007	**0.9605**	**0.023**	14.4	3.34	12	5.05	52.4	32.9	3.99	1.06	80.71	68.54	20.33	12.26
O-GT	**0.9799**	**0.005**	0.9574	0.026	**14.4**	**3.34**	**12**	**5.05**	**52.4**	**32.9**	**3.99**	**1.06**	**80.71**	**68.54**	**482.77**	**94.26**
FS-PD-FDT	0.9644	0.005	0.962	0.022	14.7	3.13	13.5	7.9	45.7	39.2	3.11	0.67	88.74	106.48	1.2	0.59
S [*]	**0.9649**	**0.005**	**0.959**	**0.023**	**11.1**	**2.28**	**7.4**	**1.71**	**20.2**	**7.4**	**2.68**	**0.43**	**28.73**	**14**	**10.9**	**5.28**
S-IC	0.9649	0.005	0.959	0.023	11.3	2.63	7.5	1.72	20.6	7.26	2.7	0.42	29.76	13.74	9.5	4.55
S-DC	0.9649	0.005	0.957	0.024	11.1	2.28	7.5	1.72	20.7	7.26	2.72	0.41	29.79	13.74	6.5	3.03
S-IC-DC	0.9649	0.005	0.959	0.023	11.3	2.62	7.6	1.78	21.2	7.45	2.74	0.41	30.6	14.09	12.6	6.62
O-SW	0.9724	0.004	**0.9708**	**0.018**	11.1	2.28	7.4	1.71	20.2	7.4	2.68	0.43	28.73	14	6.067	2.83
O-GT	**0.9749**	**0.003**	0.9599	0.021	**11.1**	**2.28**	**7.4**	**1.71**	**20.2**	**7.4**	**2.68**	**0.43**	**28.73**	**14**	**464.1**	**134.84**
FS-PD-FPA	0.9647	0.005	0.962	0.018	14.7	3.13	142.8	74.89	925.5	571.39	6	1.15	2828.86	2225.9	1.15	0.47
S	0.9666	0.005	0.946	0.027	11.5	3.37	6	2	18.8	9.59	3.01	0.64	23.26	16.9	97.9	97.2
S-IC [*]	**0.9673**	**0.004**	**0.959**	**0.022**	**11.9**	**2.92**	**5.7**	**2.06**	**17.1**	**9.93**	**2.82**	**0.71**	**19.45**	**16.68**	**52.9**	**42.49**
S-DC	0.9661	0.005	0.95	0.022	11.9	3.25	6.8	1.75	22.9	9.63	3.23	0.77	30.82	17.08	69.5	69.98
S-IC-DC	0.9665	0.004	0.952	0.021	11.7	2.98	6.7	1.83	21.7	9.82	3.09	0.77	27.85	17.56	159.2	147.13
O-SW	0.971	0.003	**0.9634**	**0.028**	11.9	2.92	5.7	2.06	17.1	9.93	2.82	0.71	19.45	16.68	7.67	5.81
O-GT [+]	**0.9754**	**0.004**	0.9629	0.022	**11.9**	**2.92**	**5.7**	**2.06**	**17.1**	**9.93**	**2.82**	**0.71**	**19.45**	**16.68**	**434.8**	**48.13**

Table 8 Results of the experimentation (rule-based learning, linguistic simplification, and partition optimization) (PIMA)

	ACC (training)		ACC (test)		NOI*NOL		NR		TRL		ARL		RBC		Runtime (s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
MP	**0.8183**	**0.017**	**0.754**	**0.047**	–	–	–	–	–	–	–	–	–	–	–	–
C45	**0.8368**	**0.024**	**0.7383**	**0.057**	–	–	**19.2**	**6.53**	**103.7**	**45.62**	**5.23**	**0.66**	–	–	–	–
PIMA																
HILK																
PD-FDT	0.942	0.02	0.7291	0.047	40	0	576.2	70.54	3064.5	335.16	5.33	0.14	84964.25	11134.63	167.3	21.56
S	0.8288	0.055	0.7238	0.066	35.5	6.88	51.9	37.75	232.9	187.66	4.1	0.7	6912.52	5584.9	5376.3	1850.2
O-SW	0.8349	0.05	0.7251	0.062	35.5	6.88	51.9	37.75	232.9	187.66	4.1	0.7	6912.52	5584.9	46.5	22.49
HILK++																
FS-PD-WM	0.8512	0.093	0.6585	0.085	25.3	7.21	320.6	178.54	2430.1	1528.01	7.2	0.92	17872.86	16816.1	0.8	0.26
S	0.7832	0.068	0.7121	0.053	22.2	9.67	27.2	29.24	173.9	228.9	4.9	1.85	1102.67	1682.42	403	444.27
S-IC	0.7835	0.083	0.6913	0.055	23	8.18	32.5	36.6	204.8	265.88	4.99	1.64	1227.34	1816.89	422.7	582.9
S-DC	0.7815	0.074	0.712	0.051	22.9	9.13	26.4	26.28	161.8	188.61	4.93	1.76	983.37	1380.69	382.8	453.03
S-IC-DC [*]	**0.7354**	**0.081**	**0.6769**	**0.055**	**15**	**11.9**	**19.4**	**29.2**	**111.6**	**199.68**	**3.2**	**2.3**	**563.7**	**1025.27**	**743.5**	**821**
O-SW	0.7396	0.082	0.6756	0.055	15	11.9	19.4	29.2	111.6	199.68	3.2	2.3	563.7	1025.27	9.45	14.2
O-GT	**0.7624**	**0.081**	**0.7042**	**0.06**	**15**	**11.9**	**19.4**	**29.2**	**111.6**	**199.68**	**3.2**	**2.3**	**563.7**	**1025.27**	**783.6**	**287.9**
FS-PD-FDT	0.8274	0.068	0.7289	0.054	25.3	7.21	155.6	181.6	719.4	863.5	3.96	1.007	2693.41	3486.13	19.8	33.37
S	0.7788	0.043	0.7381	0.055	15.5	12.13	21.3	28.65	78.5	121.15	2.59	1.108	2283.31	4592.05	381.8	687.45
S-IC	0.7771	0.045	0.751	0.065	15	11.77	20.6	28.83	68.6	107.66	2.39	0.96	2023.46	4517.47	357.9	640.99
S-DC	0.7816	0.042	0.7355	0.054	15.8	12.1	22.8	30.44	91.5	140	2.77	1.28	2152.03	4415.12	331.8	580.2
S-IC-DC [*]	**0.75**	**0.038**	**0.7263**	**0.06**	**11.2**	**8.97**	**8.9**	**7.38**	**23.5**	**27.02**	**2.05**	**0.85**	**268.28**	**563.33**	**676.5**	**1145.99**
O-SW	0.7604	0.04	**0.738**	**0.064**	11.2	8.97	8.9	7.38	23.5	27.02	2.05	0.85	268.28	563.33	12.3	15.9
O-GT	**0.768**	**0.045**	0.7336	0.06	**11.2**	**8.97**	**8.9**	**7.38**	**23.5**	**27.02**	**2.05**	**0.85**	**268.28**	**563.33**	**781.33**	**217.25**
FS-PD-FPA	0.7234	0.111	0.6311	0.139	25.3	7.21	277.5	108.78	2063.9	933.36	7.2	0.92	13704.9	9832.7	8.8	9.3
S [*]	**0.6657**	**0.098**	**0.6494**	**0.1**	**9.8**	**6.2**	**3.1**	**1.2**	**8.5**	**6.7**	**2.46**	**1.34**	**22.44**	**1.34**	**178.2**	**136.97**
S-IC	0.6648	0.11	0.6378	0.11	11.2	8.7	4.5	2.8	14.8	17.2	2.49	1.48	42.1	75.2	152.1	88.3
S-DC	0.6627	0.11	0.6378	0.14	11.4	8.3	3.8	2.4	12.2	11.8	2.74	1.63	33.57	39.12	152.2	96.8
S-IC-DC	0.6255	0.12	0.6	0.15	9.5	9.2	4.1	3.2	12.2	14.3	2.18	1.7	48.82	64.91	301.9	191.7
O-SW	0.6685	0.099	0.6494	0.1	9.8	6.2	3.1	1.2	8.5	6.7	2.46	1.34	22.44	1.34	5.15	7.35
O-GT [+]	**0.7552**	**0.021**	**0.7349**	**0.054**	**9.8**	**6.2**	**3.1**	**1.2**	**8.5**	**6.7**	**2.46**	**1.34**	**22.44**	**1.34**	**865.7**	**147.9**

were simplified without altering the initial ranking (S), and finally fuzzy partitions were adjusted by genetic tuning (O-GT). This solution is really good from the interpretability point of view and its accuracy is comparable to the one obtained by C4.5. The loss of accuracy is negligible (smaller than 2% for training and smaller than 1% regarding test). Notice that, the simplicity of the problem does not let us appreciate the effect of the different simplification strategies. All of them yield the same results given a rule induction algorithm.

In addition, O-GT and O-SW yield very similar results. Of course, O-GT is able to achieve slightly higher accuracy but at the cost of a much larger computational time. As it can be easily appreciated the runtime is always in the range of seconds except for O-GT that consumes about 300 s (5 min). This is due to the fact that we have set the same default parameters for O-GT in all the six problems. Nevertheless, in the case of IRIS it may be enough with a smaller population and a smaller number of generations what should reduce dramatically the computational time while keeping the high accuracy. On the other hand, we can see the benefits derived from the feature selection stage by making a comparison between solutions starting with and without FS. Observe how PD-FDT-S-O-SW yields the largest RBC even after simplification. Of course, it also yields the largest ACC (training). Anyway, the increase of accuracy is not so big as the decrease of interpretability when disregarding FS.

As deduced from Table 1, the WINE problem is more complex. It counts with three classes like IRIS, but the number of attributes is higher (13 against 4). Anyway, achieved results (look at Table 4) were also very encouraging. The best solution was FS-PD-FDT-S-DC-O-GT. Nevertheless, FS-PD-WM-S-IC-O-GT and FS-PD-FPA-S-IC-O-GT exhibited also very good trade-offs between interpretability and accuracy. Like in the case of IRIS, the gain of interpretability for WINE is obtained at the cost of some loss of accuracy. Anyway, it is around 3% in the worst case what is perfectly acceptable. Again, like in the IRIS problem, the runtime is kept under 10 s for all the modeling stages except for O-GT which yields about 10 min.

However, when dealing with a more complex problem like GLASS (look at Table 1), results are not so promising. Although the number of attributes (nine) is smaller than in the case of WINE, the number of instances and classes is bigger. GLASS identifies six different classes quite unbalanced. The distribution of instances per class is the following: G1 (32.71%), G2 (35.51%), G3 (7.94%), G4 (6.074%), G5 (4.205%), and G6 (13.561%).

Looking at Table 5, we see how C4.5 is able to yield a very good solution regarding both accuracy and interpretability for GLASS. ACC (test) is slightly smaller for C4.5 than MP, but ACC (training) is even higher for C4.5. Furthermore, C4.5 yields a quite compact solution with a small NR. The simplest solution, i.e., the one yielding the smallest RBC, provided by HILK++ was FS-PD-FDT-S-DC-O-GT. It is more interpretable than C4.5 in terms of TRL and ARL, although NR is larger. Notice that, FS-PD-FPA-S-IC-O-GT gets even smaller NR than C4.5, but RBC is larger because of the huge increment in ARL. Of course, looking at NOI*NOL we observe how the simplification stage affects not only to the rule base but also to the fuzzy partitions. Unfortunately, the reduction of NR and TRL is achieved at the cost of a strong reduction of accuracy. Hence, such solution is not competitive against C4.5.

In the GLASS problem, the simplest solution is also the most accurate one. It is comparable to the solution reported by MP. Nevertheless, its accuracy is not so good in comparison with C4.5. Although our solution overcomes C4.5 regarding test, the loss of accuracy is about 10% with respect to training patterns. Of course, it is not easy to establish which solution is the most interpretable one. Our method yields more rules but they are shorter than the ones provided by C4.5.

The reason why our methodology is not achieving solutions for GLASS as good as the ones obtained for the previous problems arises from the feature selection stage. The proposed method based on C4.5 is not effective in the GLASS problem since it selects 8.7 inputs (in average) out of the initial nine inputs (look at Table 2). In consequence, rule induction algorithms generate a huge number of rules which remains big even after simplification. In the future, to tackle with complex problems we should explore the introduction of a pre-processing stage based on other feature selection techniques with the aim of reducing the combinatorial rule explosion effect.

Regarding the runtime, GLASS shows how the execution time, for both simplification and optimization stages, depends a lot on the complexity of the knowledge base under consideration. In the worst case, the runtime for simplification rises up to about 2 min while it is about 23 min for optimization. Hence, the larger RBC is the larger runtime becomes. This result hampers the applicability of HILK++ to complex problems available in the UCI machine learning repository. Fortunately, the main advantage of HILK++ lies in the fact that induced knowledge is represented in a highly interpretable manner what makes possible to combine it with expert knowledge in an easy way. As a result, HILK++ is especially useful to tackle with complex real-world problems where both expert and induced knowledge are available.

Tables 6 and 7 contain the average results for two well-known benchmark classification problems in the domain of medical diagnosis (NEWTHYROID and WBCD). Reported results are similar to those achieved in the IRIS and

WINE problems in terms of accuracy and runtime (except for the simplification stage in the case of WBCD that yields execution time similar to that computed for GLASS). HILK++ clearly overwhelms C4.5 for all the interpretability indicators while accuracy is comparable: ACC(training) provided by our method is a bit smaller than the one achieved by C4.5, but HILK++ is even better regarding ACC (test) in some cases.

FS-PD-FPA-S-IC-DC-O-GT is the simplest solution for NEWTHYROID. It also yields the highest ACC (test). In the case of WBCD, the simplest solution also comes from FPA but combined with a different simplification option, S-IC instead of S-IC-DC. Results provided by FS-PD-FPA-S-IC-O-GT are especially good. In comparison with C4.5, ACC (training) is almost the same while ACC (test) is higher with our method. Results are more impressive regarding interpretability. The reduction ratio is around 46% for NR, 60% for TRL, and 28% for ARL.

Although WBCD has the same number of inputs (nine) than GLASS with three times more instances (look at Table 2), our method is able to find out a really good trade-off between accuracy and interpretability. Notice that, in this problem the feature selection process is quite effective keeping, in average, only 5.9 inputs and 2.46 linguistic terms per input (look at Table 2). As a result, the two last stages of our method (simplification and optimization) deal with compact sets of short rules. In some cases, NR is quite large before simplification what yields high runtime (up to 2.40 min in the worst case). For instance, it equals 142.8 for FPA. However, the main difference between GLASS and WBCD turns up when looking at the number of classes. WBCD includes only two classes (against the six classes of GLASS) what makes easier the simplification task which becomes much more effective.

We have also tested our method in the PIMA problem. Results are summarized at Table 8. Apparently, PIMA is a problem quite similar to WBCD. Both have only two classes, PIMA has eight inputs (one less than WBCD), and the number of instances is very close (around 700). Nevertheless, PIMA is much more complex than WBCD as deduced from the low accuracy achieved by MP and C4.5. ACC (training) is under 0.84 and ACC (test) is under 0.76. Notice that this situation is similar to GLASS where MP and C4.5 also provide low accuracy.

Again, like in GLASS, the feature selection procedure yields bad results. It selects, in average, 7.2 inputs from the eight initial ones. Then, rule induction algorithms produce hundreds of rules. It is remarkable the extremely good results in terms of interpretability provided by FS-PD-FPA-S. For instance, NR drops from 19.2 to 3.1 what means a reduction of more than 80%. The improvement is also very impressive with respect to TRL and ARL. Of course, such improvement implies losing about 6% in ACC (training). Although O-SW only produces a negligible increase of accuracy, the increment obtained with O-GT is very significant (about 9%) regarding both training and test. In consequence, we can say that sometimes it is worthy spending several minutes (almost 15 min in this case) with O-GT because it can become a very profitable investment. In addition, it is mandatory to mention that SD gets very high values for many interpretability indicators in the case of PIMA. This behavior is especially significant when considering FDT. Notice that, we have checked carefully that such values are properly computed and the abnormal values are due to the huge diversity existing among the ten folds automatically generated for cross-validation. As a result, FDT produces sets of rules of very different size and complexity what has an important impact in the rest of the modeling process.

Table 9 gives an overview on the simplest solutions, those remarked with symbol [*] in Tables 3, 4, 5, 6, 7, 8 obtained for each problem regarding both the rule induction method and the simplification strategy. The simplest solution (called Winner) for each problem is set in bold and remarked with **.

Table 9 Summary of the configurations (rule induction and simplification stages) yielding the simplest solutions

	S	S-IC	S-DC	S-IC-DC	Winners
WM	IRIS	WINE		PIMA	0
		GLASS			
		NEWTHYROID			
		WBCD			
FDT	****IRIS****		****WINE****	PIMA	3
	NEWTHYROID		****GLASS****		
	WBCD				
FPA	IRIS	WINE		****NEWTHYROID****	3
	****PIMA****	GLASS			
		****WBCD****			
Total	6	7	2	3	
Winners	2	1	2	1	

Making a comparison among the proposed rule ranking options, at first glance S-IC seems to be the best one because it gives the best results in most cases (seven over eighteen). Nevertheless, we observe that many times (six over eighteen) preserving the rule ranking provided by rule induction (S) yields better results. On the other hand, S-DC and S-IC-DC only yield the simplest solution in two and three cases each. However, the two cases in which S-DC produces the simplest solution correspond to two out of the six winner cases.

Looking at the rule induction technique, the best choice depends on the application context. We observe that FDT gives the best solutions for the three general purpose classification problems (IRIS, WINE, and GLASS). In turn, FPA yields the best solutions for the three remaining problems (NEWTHYROID, WBCD, and PIMA), those related to medical diagnosis. We should discard WM because it produces a lot of very specific and long rules with low generalization ability for all the six problems under study.

It is interesting to observe how S-IC seems the preferred strategy for most problems when looking at WM and FPA, but it is never selected in combination with FDT. This is due to the intrinsic rule nature that characterizes each rule induction method. WM and FPA produce quite specific complete rules (including all inputs). As a result, all rules have the same initial complexity and in such situation S-IC becomes the best strategy. Nevertheless, FDT generates more general incomplete rules where only a subset of inputs is considered. Therefore, different rules have different complexity depending on their length. In consequence, S-DC and S-IC-DC are able to get better results. Thus, we can draw the next conclusion: The best simplification strategy depends on the rule induction technique, but also on the specific problem under consideration. Notice that, we would like to remark that the simplification process exhibits an implicit memory effect because every simplification step has an influence in the whole series of subsequent ones. Since S-IC-DC only analyzes one step, we guess it may achieve even better results keeping a temporal sliding window, but it would increase significantly the computational cost. Hence, it remains an open research challenge.

Finally, Table 10 shows a comparison between C4.5 and the best solutions provided by HILK++, those remarked with symbol [+] in Tables 3, 4, 5, 6, 7, 8. We have computed the difference (Diff) between C4.5 and HILK++ in such a way that Diff greater than zero means C4.5 gets larger values than HILK++ for the selected quality indicators. Therefore, a positive Diff regarding accuracy means C4.5 overcomes HILK++. On the contrary, a positive Diff in terms of interpretability implies HILK++ is able to achieve a more compact solution than C4.5. In all data sets, our method achieves smaller ACC (training) than C4.5, but it yields almost the same ACC (test). Moreover, HILK++ provides slightly bigger ACC (test) than C4.5 in three of the analyzed problems. This result remarks the good generalization ability exhibited by our method. It is translating the specific rules automatically extracted from data (by means of well-known rule induction techniques) into quite general ones much closer to those rules that would be defined by an expert. With respect to interpretability our method is definitely powerful. In most problems, HILK++ clearly overwhelms C4.5 regarding all the three indicators. Only in the GLASS problem HILK++ produces bigger NR but it is compensated with much smaller TRL and ARL.

Table 10 Comparison between C4.5 and the best solutions provided by HILK++ (Diff = C4.5 − HILK++)

Dataset	ACC (training)	ACC (test)	NR	TRL	ARL
IRIS	0.0191	0.0089	1.7	9.5	1.635
WINE	0.0255	0.0266	1.1	4.2	0.363
GLASS	0.1005	−0.0024	−7.6	33.8	2.53
NEWTHYROID	0.0294	−0.0003	3.4	19.3	1.344
WBCD	0.0047	−0.0025	4.8	24.8	1.07
PIMA	0.0816	0.0034	16.1	95.2	2.77
Mean	0.0435	0.0056	3.25	31.13	1.619
SD	0.0383	0.0111	7.65	33.12	0.9

4 Conclusions

This paper has proposed new functionalities (C45-based feature selection and interpretability-guided rule ranking) as well as a new way of combining tools (partition design, rule generation, simplification, and optimization) provided by the HILK methodology. In consequence, HILK has been upgraded to a new and powerful HILK++ able to get very encouraging results.

On the one hand, at first glance the overall methodology may appear quite complex due to the availability of several alternative algorithms for each main modeling stage. Nevertheless, this fact should be seen as one of the main advantages of HILK++. In practice, methodologies with a small number of parameters to configure are very attractive for non-expert users because they are easily applicable. Unfortunately, default configurations usually do not work properly when looking for personalized solutions to complex real-world problems. The best configuration of HILK++ will depend on each specific problem under consideration but it is important to remark that HILK++ is flexible enough (thanks to the combination of several algorithms in each step) to be easily adaptable to a wide range of problems. Of course, based on the achieved experimental results we may suggest a default configuration for HILK++. Let us propose using FS-PD-FDT-S-DC-O-SW for general purpose classification problems. However, we have seen that considering FPA may yield better results when dealing with medical diagnosis problems. Anyway, if the default configuration does not give good results for a specific problem then we should try the alternatives configurations.

On the other hand, although the main ability of HILK consisted in combining expert and induced knowledge in an interpretable and human-oriented way under the same fuzzy framework, this work has focused only on knowledge automatically generated from data. As a result, we are exploiting only a small part of the HILK power. Nonetheless, this way of working let us make a fair comparison between HILK++ and other pure machine learning techniques available in the literature. Furthermore, the more interpretable the automatically generated knowledge base is the easier the understanding of its linguistic description becomes, because it is closer to the natural language usually managed by human beings. In consequence, the combination between expert and induced knowledge would be more easy and effective.

In most analyzed problems, our method is able to yield classifiers more robust, compact and comprehensible than the ones obtained with C4.5. However, such gain of interpretability is obtained at the cost of a loss of accuracy. Comparing C4.5 with the simplest classifiers obtained by our method, we lose, in average (for the six problems), around 4.35% regarding training patterns and only around 0.6% with respect to test. This reduction of accuracy seems absolutely reasonable taking into account the simplicity and comprehensibility of the classifiers designed. It is very remarkable the huge reduction of TRL that is (in average) higher than 30. In addition, our method exhibits a good generalization ability obtaining similar accuracy for both training and test. This fact is mainly due to the effective combination of the simplification and optimization procedures.

Regarding the computational cost, the experimental analysis showed how the largest overhead on runtime is produced because of the simplification and optimization stages which may take up to several minutes. It varies a lot depending on the complexity of the knowledge base under consideration. Of course, such complexity depends on the selected rule induction technique but also on the number of attributes and instances included in the data set. Anyway, it is kept reasonably small for the whole modeling process in the six analyzed problems, except for the O-GT strategy which is usually quite large. Hence, if the execution time spent in the generation of the model is a critical pre-requisite in our application, then we may discard the use of O-GT because O-SW is often able to provide very similar results with a dramatically smaller runtime in most problems. However, if we have time enough we can trust on O-GT. It usually yields accuracy only slightly larger than O-SW, but sometimes the improvement can be very significant like in the PIMA problem. The most difficult task related to O-GT is how to set the most suitable parameters of the algorithm with the aim of achieving a good trade-off between accuracy and runtime. Of course, spending infinite time does not guarantee to find the global optimum due to the random nature of genetic tuning. Nevertheless, we have to admit that adapting the configuration parameters to the specific characteristics of each problem is likely to reduce significantly the overall runtime.

In comparison with the previous version of HILK, the main improvements are obtained thanks to the addition of the new feature selection stage. It finds out the most significant variables but also the most suitable number of terms per input. In consequence, rule induction techniques produce a more compact initial rule set what makes easier and much more efficient and effective the subsequent simplification and optimization stages. It is especially important for the scalability of our methodology.

However, more work still remains to be done in order to extend our method to be successfully applied when dealing with more complex problems. As future work, we will incorporate the characteristic ability of HILK for combining expert and induced knowledge into HILK++. Furthermore, as a complement to the feature selection process we observe that it would be nice exploring other voting techniques for selecting partitions as part of the fuzzy partition design. Moreover, we should analyze the chance of adding a pre-processing stage based on other feature selection techniques.

Acknowledgments This work has been funded by the Foundation for the Advancement of Soft Computing located at Mieres (Asturias, Spain) and the Spanish Government (CICYT) under project: TIN2008-06890-C02-01.

References

Abonyi J, Roubos JA, Szeifert F (2003) Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. Int J Approx Reason 32:1–21

Alcalá R, Alcalá-Fdez J, Gacto MJ, Herrera F (2007) Rule base reduction and genetic tuning of fuzzy systems based on the linguistic 3-tuples representation. Soft Comput 11(5):401–419

Alonso JM, Cordón O, Guillaume S, Magdalena L (2007) Highly interpretable linguistic knowledge bases optimization: genetic tuning versus Solis-Wetts. Looking for a good interpretability–accuracy trade-off. In: Annual IEEE international conference on fuzzy systems, London, UK, pp 901–906

Alonso JM, Magdalena L, Guillaume S (2008) HILK: a new methodology for designing highly interpretable linguistic knowledge bases using the fuzzy logic formalism. Int J Intell Syst 23(7):761–794

Alonso JM, Magdalena L, González-Rodríguez G (2009) Looking for a good fuzzy system interpretability index: an experimental approach. Int J Approx Reason 51:115–134

Antonelli M, Ducange P, Lazzerini B, Marcelloni F (2009) Exploiting a new interpretability index in the multi-objective evolutionary learning of mamdani fuzzy rule-based systems. In: Ninth international conference on intelligent system design and applications, IEEE, Pisa, Italy, pp 115–120

Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York

Casillas J, Cordón O, Herrera F, Magdalena L (2003a) Accuracy improvements in linguistic fuzzy modeling, vol 129. Studies in fuzziness and soft computing. Springer, Heidelberg

Casillas J, Cordón O, Herrera F, Magdalena L (2003b) Interpretability issues in fuzzy modeling, vol 128. Studies in fuzziness and soft computing. Springer, Heidelberg

Casillas J, Cordón O, Del Jesús MJ, Herrera F (2005) Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. IEEE Trans Fuzzy Syst 13:13–29

Castro JL (1995) Fuzzy logic controllers are universal approximators. IEEE Trans Syst Man Cybern 25(4):629–635

Chen MY (2002) Establishing interpretable fuzzy models from numeric data. In: 4th world congress on intelligent control and automation IEEE, pp 1857–1861

Cordón O, Herrera F (1997) A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples. Int J Approx Reason 17(4):369–407

Cordón O, Herrera F, Hoffmann F, Magdalena L (2001) Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases, vol 19. Advances in fuzzy systems—applications and theory. World Scientific Publishing Co. Pvt. Ltd., Singapore

Cordón O, Gomide F, Herrera F, Hoffmann F, Magdalena L (2004) Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets Syst 141:5–31

De Oliveira JV (1999) Semantic constraints for membership function optimization. IEEE Trans Syst Man Cybern A Syst Hum 29(1):128–138

Gacto MJ, Alcalá R, Herrera F (2010) Integration of an index to preserve the semantic interpretability in the multi-objective evolutionary rule selection and tuning of linguistic fuzzy systems. IEEE Trans Fuzzy Syst. doi:10.1109/TFUZZ.2010.2041008

Glorennec PY (1999) Algorithmes d'apprentissage pour systèmes d'inférence floue. Editions Hermès, Paris

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, New York

Guillaume S (2001) Designing fuzzy inference systems from data: an interpretability-oriented review. IEEE Trans Fuzzy Syst 9(3):426–443

Guillaume S, Charnomordic B (2004) Generating an interpretable family of fuzzy partitions. IEEE Trans Fuzzy Syst 12(3):324–335

Hartigan JA, Wong MA (1979) A k-means clustering algorithm. Appl Stat 28:100–108

Herrera F (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. Evol Intell 1(1):27–46

Hjorth JSU (1994) Computer intensive statistical methods validation, model selection, and bootstrap. Chapman & Hall, London

Hüllermeier E (2005) Fuzzy methods in machine learning and data mining: status and prospects. Fuzzy Sets Syst 156:387–406

Ichihashi H, Shirai T, Nagasaka K, Miyoshi T (1996) Neuro-fuzzy ID3: a method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning. Fuzzy Sets Syst 81:157–167

Ishibuchi H, Nojima Y (2007) Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. Int J Approx Reason 44:4–31

Karr CL (1991) Genetic algorithms for fuzzy controllers. AI Expert 6(2):26–33

Loquin K, Strauss O (2006) Fuzzy histograms and density estimation. Adv Soft Comput 6:45–52

Mamdani EH (1977) Application of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Trans Comput 26(12):1182–1191

Mencar C, Fanelli AM (2008) Interpretability constraints for fuzzy information granulation. Inf Sci 178:4585–4618

Miller GA (1956) The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol Rev 101(2):343–352

Plutowski M, Sakata S, White H (1994) Cross-validation estimates IMSE. In: Cowan JD, Tesauro G, Alspector J (eds.) Advances in neural information processing systems 6. Morgan Kaufman, San Mateo, pp 391–398

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Mateo

Quinlan JR (1996) Improved use of continuous attributes in C4.5. J Artif Intell Res 4:77–90

Ruspini EH (1969) A new approach to clustering. Inf Control 15(1):22–32

Saaty TL, Ozdemir MS (2003) Why the magic number seven plus or minus two. Math Comput Model 38:233–244

Solis FJ, Wets RJB (1981) Minimization by random search techniques. Math Oper Res 6:19–30

Van Broekhoven E, Adriaenssens V, De Baets B (2007) Interpretability-preserving genetic optimization of linguistic terms in fuzzy models for fuzzy ordered classification: an ecological case study. Int J Approx Reason 44:65–90

Wang LX, Mendel JM (1992) Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 22(6):1414–1427

Zadeh LA (1975) The concept of a linguistic variable and its application to approximate reasoning. Part I. Inf Sci 8:199–249