Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems 159 (2008) 3201 – 3220

FUZZY sets and systems

www.elsevier.com/locate/fss

A methodology for automated fuzzy model generation

Markos G. Tsipouras[a], Themis P. Exarchos[a, b], Dimitrios I. Fotiadis[a, c, d, *]

[a]*Department of Computer Science, Unit of Medical Technology and Intelligent Information Systems, University of Ioannina, GR 45110 Ioannina, Greece*

[b]*Department of Medical Physics, Medical School, University of Ioannina, GR 45110 Ioannina, Greece*

[c]*Biomedical Research Institute – FORTH, GR 45110 Ioannina, Greece*

[d]*Michaelideion Cardiology Center, GR 45110 Ioannina, Greece*

Received 19 August 2007; received in revised form 13 March 2008; accepted 5 April 2008
Available online 12 April 2008

Abstract

In this paper we propose a generic methodology for the automated generation of fuzzy models. The methodology is realized in three stages. Initially, a crisp model is created and in the second stage it is transformed to a fuzzy one. In the third stage, all parameters entering the fuzzy model are optimized. The proposed methodology is novel and generic since it can integrate alternative techniques in each of its stages. A specific realization of this methodology is implemented, using decision trees for the creation of the crisp model, the sigmoid function, the min–max operators and the maximum defuzzifier, for the transformation of the crisp model into a fuzzy one, and four different optimization strategies, including global and local optimization techniques, as well as, hybrid approaches. The proposed methodology presents several advantages and novelties: the transformation of the crisp model to the respective fuzzy one is straightforward ensuring its full automated nature and it introduces a set of parameters, expressing the importance of each fuzzy rule. The above realization is extensively evaluated using several benchmark data sets from the UCI machine learning repository and the obtained results indicate its high efficiency.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Decision trees; Fuzzy modeling; Optimization; Weighted fuzzy rules

1. Introduction

Fuzzy logic is the extension of the classical crisp (binary) logic into a multivariate form. Fuzzy logic is closer to human logic, thus it can deal with real-world noisy and imprecise data [30,29]. Fuzzy models experience several advantages compared to crisp ones. The main one is that they have more flexible decision boundaries, and thus, they are characterized by their higher ability to adjust to a specific domain of application and more accurately reflect its particularities. A fuzzy model can be created by defining an initial crisp model (set of rules) and then fuzzifying it. This approach is a complex task, since several issues must be resolved for the fuzzy model to be generated. First, the origin of the rules must be defined, which determines the philosophy of the method; if experts' knowledge was used then the generated fuzzy model will be knowledge-based, while if data-mining techniques were employed then a data-driven fuzzy model will be generated. In the fuzzification step, there are certain fundamental features which must be

* Corresponding author at: Department of Computer Science, Unit of Medical Technology and Intelligent Information Systems, University of Ioannina, PO Box 1186, GR 451 10 Ioannina, Greece. Tel.: +30 26510 98803; fax: +30 26510 98889.
E-mail address: fotiadis@cs.uoi.gr (D.I. Fotiadis).

0165-0114/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2008.04.004

provided in order to create the fuzzy model, such as: the fuzzy membership function, the fuzzy operators, the defuzzification approach and the use of weights. Following this approach, the generated fuzzy model resembles the decision making processes of the initial crisp model and thus its parameters must be tuned. This "tuning'' can be performed using parameter optimization.

Several approaches have been proposed in the literature for the development of knowledge-based fuzzy models. In most of them, the model is trained using a known optimization technique i.e. fuzzy rules with genetic algorithms [18], fuzzy rules with simulated annealing [9], multicriteria decision analysis with genetic algorithms [10], fuzzy rules with modified controlled random search [28], etc. Also, several research attempts exist in the literature, which integrate data-mining techniques with fuzzy modeling. More specifically, the presented approaches can be classified into three main categories: (i) induction of a crisp decision tree from the data and then its fuzzification, resulting into a fuzzy decision tree [14,23,17,5,7], (ii) induction of a fuzzy decision tree, integrating fuzzy techniques during the tree construction [34,12,2,13,31,25], (iii) induction of a crisp decision tree, extraction of a set of rules from it and fuzzification of these rules [1]. In the first category, Jeng et al. [14] proposed the integration of fuzzy theory into the regular inductive learning method for single dimension decision problems. Suarez and Lutsko [23] proposed a fuzzification of a CART decision tree. Olaru et al. [17] proposed the fuzzification of a crisp regression tree. Chen and Jeng [5] extended the method proposed in [14], integrating fuzzy theory into the regular inductive learning method for multidimensional decision problems. Crockett et al. [7] constructed fuzzy decision trees, fuzzifying crisp decision trees induced from the data using the C4.5 algorithm. In the second category, Yuan and Shaw [34] introduced a heuristic algorithm for generating fuzzy decision trees, similar to the ID3 method, based on the measurement of the classification ability. Ichihashi et al. [12] proposed a method of inducing fuzzy decision trees, based on the fuzzy ID3 algorithm, using expert's partial knowledge. Apolloni et al. [2] presented a method for learning fuzzy decision trees, using recurrent neural networks to suggest the next move during the descent along the branches of the tree. Janikow [13] provided a detailed investigation for fuzzy decision trees, combining fuzzy representation with symbolic decision trees. Wang et al. [31] investigated the optimization of fuzzy decision trees and proposed a branch-merging algorithm for FDT generation. Tsang et al. [25] proposed a methodology to improve the learning accuracy of fuzzy decision trees, using hybrid neural networks. Concerning the third category, Abonyi et al. [1] proposed a method for fuzzification of rules extracted from decision trees, induced by the C4.5 algorithm. They optimized and simplified the fuzzy set of rules using genetic algorithms.

In most of these approaches, the ID3 tree-induction algorithm is employed, while in some articles the CART or the C4.5 tree-induction algorithms are used. Fuzzy modeling has been treated with several different approaches concerning the fuzzification of the input variables, the construction of the inference engine and the defuzzification procedure. Different approaches for the fuzzy operators have been employed. All approaches include an optimization stage in order to tune the parameters entering the fuzzy models to fit a specific problem, described by a data set. The majority of the published articles employ genetic algorithms in this stage.

All the above methodologies are based on decision trees, thus they might not be applicable when a different rule-mining technique is employed or rules based on experts' knowledge are available. Furthermore, since the ID3 tree-induction algorithm has several limitations [17,7], all methodologies based on ID3 tree-induction algorithm inherit these limitations. In addition, fuzzy decision trees, created either by the fuzzification of a crisp decision tree or by the integration of fuzzy modeling during the tree construction, may be inconsistent or incomplete [13]. Finally, in several research attempts the evaluation is not adequate, since it is based on one or a small number of data sets.

Concerning use of weights in fuzzy modeling, two approaches have been proposed: (i) local weights, which are used to indicate the relative degree of importance of a proposition contributing to its consequent, thus one local weight is assigned to each fuzzy conjunct. Local weights play an important role in many real-world problems. For example, in medical diagnostic systems it is common to observe that a particular symptom combined with other symptoms may lead to a possible disease and thus it is important to assign a local weight to each symptom in order to show the relative degree (weight) of each symptom leading to the consequent (a disease) [4,11,26]; (ii) global weights, which are used to represent the relative degree of importance of each rule's contribution, thus one global weight is assigned to each fuzzy rule [33].

In this work, we propose a methodology for automated fuzzy model generation which includes three stages: (i) crisp model extraction, (ii) fuzzy model creation and (iii) parameter optimization. In the first stage, a crisp model is created; depending on the approach, which will be employed for the crisp model creation, the methodology can be knowledge-based, if the initial crisp model is defined by experts, or data-driven, if the initial crisp model is extracted from the available data, or hybrid, if a combination of the above is employed. In the second stage, this crisp model is transformed to the corresponding fuzzy model; several new parameters are introduced due to the fuzzification of the decision boundaries. In addition, a set of weights is employed, each of them representing the relative importance of each class (class weights). Finally (in the third stage), optimization is performed in order to tune all the parameters entering the fuzzy model. The methodology presents several advantages and novelties: it is generic since it is not based on a specific technique for crisp model generation; expert knowledge or any rule-mining technique can be adapted to generate the crisp model. This offers flexibility, since the methodology does not depend on the domain of application. This is advantageous since it can integrate state-of-the-art rule-mining methods, as well as future developments or even hybrid approaches, combining expert knowledge with the mined knowledge. Also, different approaches concerning the elements of the fuzzy model and alternative optimization techniques can be integrated. Another major advantage of the methodology is that the transformation of the crisp model to the respective fuzzy one is straightforward, ensuring in this way the fully automated nature of the methodology. Finally, the introduction of class weights is a novel feature which allows the fuzzy model to be more flexible and adaptable.

In this paper we present also a specific realization of the methodology. The methodology is applied in its data-driven form (i.e. the initial crisp model is generated from an annotated data set, using data-mining techniques), since for several of the domains that we applied or methodology, experts' rules do not exist. The crisp rules are extracted from a decision tree, induced by the data. These rules constitute a crisp model, which has axis-parallel decision boundaries. In order to create non-axis-parallel decision boundaries, the crisp model is transformed to a fuzzy model, replacing all the fundamental elements of the crisp model (crisp membership function, binary AND and OR operators and decision function) with fuzzy equivalents (fuzzy membership function, $T_{\rm norm}$ and $S_{\rm norm}$ functions and defuzzification function); specific realizations of these steps are demonstrated. The above, introduces several new parameters into the fuzzy model (compared to the crisp one), as well as the class weights. Optimization is performed for all parameters. Global, local and hybrid optimization strategies are tested. A large number of benchmark classification data sets is used for the evaluation and the reported results indicate high classification accuracy, in the majority of cases superior to the respective accuracy of the crisp model. Also, the generation of fuzzy models, based on the fuzzification and optimization of a set of rules extracted from the decision tree, instead of a fuzzy decision tree (largely proposed in the literature), is a novel feature which offers more flexibility and adaptation ability to a specific data set, while keeping the complexity of the decision making process the same.

The manuscript is structured as follows: initially, the proposed methodology is described (Section 2), while the selections which are made for the specific realization and techniques employed in each stage are analyzed in Section 3. A brief description of the data sets, a detailed presentation of the obtained results and a comparison with standard classification algorithms are provided in Section 4. Finally, an extensive discussion of the obtained results along with a detailed comparison with other researchers' results presented in the literature, obtained by similar approaches, are given in Section 5.

2. Methodology

A flowchart of the methodology is shown in Fig. 1.

2.1. Crisp model creation

An initial set of crisp rules is used to create a crisp rule-based classifier (crisp model), which is a classification technique, based on a collection of "if…then…'' rules. Each crisp classification rule $r^{\rm c}_{i,j}(x, \theta^{\rm c}_{i,j})$ is expressed as $r^{\rm c}_{i,j}(x, \theta^{\rm c}_{i,j})$: $(Cond^{\rm c}_{i,j}(x, \theta^{\rm c}_{i,j})) \rightarrow y_i$, where x is a pattern characterized by the features a_l, $l = 1, \ldots, n_{\rm f}$ $(x = [a_1, a_2, \ldots, a_{n_{\rm f}}]$, $n_{\rm f}$ is the number of features), $i = 1, \ldots, I$, denotes the class that the crisp classification rule predicts (I is the total number of classes), $j = 1, \ldots, J_i$, denotes the classification rule which predicts the ith class (J_i is the total number of classification rules that predict the ith class). $Cond^{\rm c}_{i,j}$ is the crisp rule's antecedent or precondition, containing a conjunction of feature tests (conjuncts): $Cond^{\rm c}_{i,j}(x, \theta^{\rm c}_{i,j}) = g^{\rm c}(a_{i,j,1}, \theta^{\rm c}_{i,j,1}) \wedge g^{\rm c}(a_{i,j,2}, \theta^{\rm c}_{i,j,2}) \wedge \cdots \wedge g^{\rm c}(a_{i,j,K_{i,j}}, \theta^{\rm c}_{i,j,K_{i,j}})$, where $(a, \theta^{\rm c})$ is a feature–parameter pair, $g^{\rm c}$ is the crisp membership function, defined as: $g^{\rm c}(a, \theta^{\rm c}) = a\ op\ \theta^{\rm c}$, where op is a comparison operator chosen from the set $\{=, \neq, <, >, \leqslant, \geqslant\}$, symbol $\wedge$ denotes the AND binary operator and $K_{i,j}$ is the number of the conjuncts of the $r^{\rm c}_{i,j}$ crisp rule. The result of the crisp membership function and of each $r^{\rm c}_{i,j}$ rule

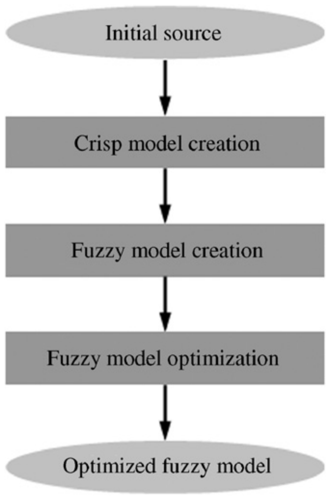

Fig. 1. Flowchart of the proposed methodology for the automated generation of fuzzy models.

is 0 or 1. The number of feature–parameter pairs is not the same for each $Cond^{\mathrm{c}}_{i,j}$ and not all features of pattern x are necessarily used in each $Cond^{\mathrm{c}}_{i,j}$. Therefore, we use the notation $a_{i,j,k}$ and $\theta^{\mathrm{c}}_{i,j,k}$ which denote the feature and the parameter, respectively, which are used in the kth conjunct of the jth classification rule that predicts the ith class. All parameters used in the $r^{\mathrm{c}}_{i,j}$ rule are included in the $\theta^{\mathrm{c}}_{i,j}$ vector ($\theta^{\mathrm{c}}_{i,j} = \{\theta^{\mathrm{c}}_{i,j,1}, \theta^{\mathrm{c}}_{i,j,2}, \ldots, \theta^{\mathrm{c}}_{i,j,K_{i,j}}\}$). Finally, y_i is the predicted class, with $y_i = i$.

All $r^{\mathrm{c}}_{i,j}$ crisp rules that have as consequent the same class are combined in a single rule (class rule), thus one class rule (R^{c}_i, $i = 1, \ldots, I$) is defined for each class i:

$$
\begin{aligned}
&R^{\mathrm{c}}_i(x, \theta^{\mathrm{c}}_i) : (r^{\mathrm{c}}_{i,1} \vee r^{\mathrm{c}}_{i,2} \vee \cdots \vee r^{\mathrm{c}}_{i,J_i}) \to y_i \\
&\Rightarrow R^{\mathrm{c}}_i(x, \theta^{\mathrm{c}}_i) : \left(\begin{array}{c}
(g^{\mathrm{c}}(a_{i,1,1}, \theta^{\mathrm{c}}_{i,1,1}) \wedge g^{\mathrm{c}}(a_{i,1,2}, \theta^{\mathrm{c}}_{i,1,2}) \wedge \cdots \wedge g^{\mathrm{c}}(a_{i,1,K_{i,j}}, \theta^{\mathrm{c}}_{i,1,K_{i,j}})) \\
\vee (g^{\mathrm{c}}(a_{i,2,1}, \theta^{\mathrm{c}}_{i,2,1}) \wedge g^{\mathrm{c}}(a_{i,2,2}, \theta^{\mathrm{c}}_{i,2,2}) \wedge \cdots \wedge g^{\mathrm{c}}(a_{i,2,K_{i,j}}, \theta^{\mathrm{c}}_{i,2,K_{i,j}})) \\
\vee \ldots, \\
(g^{\mathrm{c}}(a_{i,J_i,1}, \theta^{\mathrm{c}}_{i,J_i,1}) \wedge g^{\mathrm{c}}(a_{i,J_i,2}, \theta^{\mathrm{c}}_{i,J_i,2}) \wedge \cdots \wedge g^{\mathrm{c}}(a_{i,J_i,K_{i,j}}, \theta^{\mathrm{c}}_{i,J_i,K_{i,j}}))
\end{array} \right) \to y_i
\end{aligned} \tag{1}
$$

where $\vee$ is the OR binary operator, θ^{c}_i is a vector containing all parameters used in the R^{c}_i class rule: $\theta^{\mathrm{c}}_i = \{\theta^{\mathrm{c}}_{i,1}, \theta^{\mathrm{c}}_{i,2}, \ldots, \theta^{\mathrm{c}}_{i,J_i}\}$. Thus, the crisp model (M^{c}) consists of I crisp class rules R^{c}_i. The result of each R^{c}_i class rule is 0 or 1. This form is the disjunctive normal form (DNF). The final decision (class) of the M^{c} is made using the results from all class rules: $M^{\mathrm{c}}(x, \Theta^{\mathrm{c}}, t) = F^{\mathrm{c}}(t_1 \cdot R^{\mathrm{c}}_1, t_2 \cdot R^{\mathrm{c}}_2, \ldots, t_I \cdot R^{\mathrm{c}}_I)$, where Θ^{c} is a vector containing all parameters ($\Theta^{\mathrm{c}} = \{\theta^{\mathrm{c}}_1, \theta^{\mathrm{c}}_2, \ldots, \theta^{\mathrm{c}}_i\}$), t is a vector of weights, each one of them assigned to a class (t_i is assigned to the ith class) and F^{c} is a function combining the outcomes of all R^{c}_k crisp class rules and results to one of the classes (decision function).

The crisp class rules R^{c}_i are considered of the same "power'', i.e. each one of them is considered equally important with all others. This assumption does not accurately reflect the properties of real-world data sets where each class, and thus its respective rule, has its own importance, depending on the domain of application and the specific data set concerning this domain. To overcome it, a vector of weights t is introduced. If the initial set of rules is defined by a domain expert, then weights t may also be defined based on expert's knowledge. If rule-mining techniques are employed, then they may provide metrics reflecting a generated rule's importance, such as the coverage or the accuracy of a specific rule, which can be used to produce the weight of each crisp class rule.

2.2. Fuzzy model creation

The crisp model is based on axis-parallel decision boundaries. This is a limitation that can be treated with the fuzzification of the crisp rules, which introduces flexibility in the decision boundaries [1]. The crisp model (M^{c}) is transformed into a fuzzy model (M^{f}) as follows:

- A fuzzy membership function $g^{\mathrm{f}}(a, \theta^{\mathrm{f}})$ is used instead of the crisp $g^{\mathrm{c}}(a, \theta^{\mathrm{c}})$. In this case, θ^{f} is a vector containing all parameters used in the fuzzy membership function (and not a single parameter as θ^{c}), and its size depends on the selection of the fuzzy membership function.
- T_{norm} and S_{norm} functions are used instead of the binary AND and OR operators, respectively.
- A defuzzification function is used instead of the F^{c} function.

Thus, each fuzzy rule $r^{\mathrm{f}}_{i,j}(x, \theta^{\mathrm{f}}_{i,j}) \to y_i$ is defined as $r^{\mathrm{f}}_{i,j}(x, \theta^{\mathrm{f}}_{i,j}) = Cond^{\mathrm{f}}_{i,j}(x, \theta^{\mathrm{f}}_{i,j})$, $Cond^{\mathrm{f}}_{i,j} = T_{\mathrm{norm}}(g^{\mathrm{f}}(a_{i,j,1}, \theta^{\mathrm{f}}_{i,j,1}), g^{\mathrm{f}}(a_{i,j,2}, \theta^{\mathrm{f}}_{i,j,2}), \ldots, g^{\mathrm{f}}(a_{i,j,J_i}, \theta^{\mathrm{f}}_{i,j,J_i}))$, g^{f} is a monotonic fuzzy membership function, with $g^{\mathrm{f}}(\cdot) \in [0, 1]$. Again, $a_{i,j,k}$ and $\theta^{\mathrm{f}}_{i,j,k}$ denote the feature and the parameter vector, respectively, which are used in the kth conjunct of the jth fuzzy rule which predicts the ith class. All parameters used in the $r^{\mathrm{f}}_{i,j}$ rule are included in $\theta^{\mathrm{f}}_{i,j}(\theta^{\mathrm{f}}_{i,j} = \{\theta^{\mathrm{f}}_{i,j,1}, \theta^{\mathrm{f}}_{i,j,2}, \ldots, \theta^{\mathrm{f}}_{i,j,K_{i,j}}\})$; the size of the $\theta^{\mathrm{f}}_{r^{\mathrm{f}}_i}$ matrix is $K_{i,j} \times n_{\mathrm{p}}$ (n_{p} is the number of parameters used in the fuzzy membership function) because one vector of n_{p} parameters is used in each conjunct of the $r^{\mathrm{f}}_{i,j}$ rule. The result of the $r^{\mathrm{f}}_{i,j}$ fuzzy rule is again in the [0, 1] interval and denotes the score of the rule concerning its predicted class y_i.

Composition inference [30] is applied to all $r^{\mathrm{f}}_{i,j}$ fuzzy rules having as consequent the same class, i.e. they are combined in a single rule and thus, one fuzzy class rule ($R^{\mathrm{f}}_i(x, \theta^{\mathrm{f}}_i) \to y_i, i = 1, \ldots, I$) is defined for each class:

$$\begin{aligned} R^{\mathrm{f}}_i(x, \theta^{\mathrm{f}}_i) &= S_{\mathrm{norm}}(r^{\mathrm{f}}_{i,1}, r^{\mathrm{f}}_{i,2}, \ldots, r^{\mathrm{f}}_{i,J_i}) \\ &= S_{\mathrm{norm}}\begin{pmatrix} T_{\mathrm{norm}}(g^{\mathrm{f}}(a_{i,1,1}, \theta^{\mathrm{f}}_{i,1,1}), g^{\mathrm{f}}(a_{i,1,2}, \theta^{\mathrm{f}}_{i,1,2}), \ldots, g^{\mathrm{f}}(a_{i,1,1}, \theta^{\mathrm{f}}_{i,1,1})), \\ T_{\mathrm{norm}}(g^{\mathrm{f}}(a_{i,2,1}, \theta^{\mathrm{f}}_{i,2,1}), g^{\mathrm{f}}(a_{i,2,2}, \theta^{\mathrm{f}}_{i,2,2}), \ldots, g^{\mathrm{f}}(a_{i,1,1}, \theta^{\mathrm{f}}_{i,1,1})), \\ \ldots, \\ T_{\mathrm{norm}}(g^{\mathrm{f}}(a_{i,J_i,1}, \theta^{\mathrm{f}}_{i,J_i,1}), g^{\mathrm{f}}(a_{i,J_i,2}, \theta^{\mathrm{f}}_{i,J_i,2}), \ldots, g^{\mathrm{f}}(a_{i,J_i,K_{i,j}}, \theta^{\mathrm{f}}_{i,J_i,K_{i,j}})) \end{pmatrix}, \end{aligned} \tag{2}$$

where θ^{f}_i contains all parameters used in the R^{f}_i fuzzy class rule: $\theta^{\mathrm{f}}_i = \{\theta^{\mathrm{f}}_{i,1}, \theta^{\mathrm{f}}_{i,2}, \ldots, \theta^{\mathrm{f}}_{i,J_i}\}$ and J_i is the number of $r^{\mathrm{f}}_{i,j}$ fuzzy rules predicting the ith class. Thus, M^{f} consists of I fuzzy class rules R^{f}_i. The result of each R^{f}_i class rule is in the range [0, 1] and denotes the score of the fuzzy class rule concerning its class y_i. Finally, individual-rule inference [30] is applied to the class rules: the results from all class rules are combined in order to reach the final decision (class) of the M^{f}: $M^{\mathrm{f}}(x, \Theta^{\mathrm{f}}, t) = F^{\mathrm{f}}(t_1 \cdot R^{\mathrm{f}}_1, t_2 \cdot R^{\mathrm{f}}_2, \ldots, t_I \cdot R^{\mathrm{f}}_I)$, where Θ^{f} contains all parameters ($\Theta^{\mathrm{f}} = \{\theta^{\mathrm{f}}_1, \theta^{\mathrm{f}}_2, \ldots, \theta^{\mathrm{f}}_I\}$) and F^{f} is a defuzzification function combining the outcomes of all R^{f}_i fuzzy class rules and results to one of the classes.

The transformation of the crisp model to the respective fuzzy model greatly depends on the selection of the fuzzifier (fuzzy membership function), the T_{norm} and S_{norm} and the defuzzifier; if specific combinations among these are chosen then known solutions from the literature can be used to express the explicit mathematical input–output of the fuzzy model [30,35]. Also, the transformation is straightforward and thus, it can be performed easily in a fully automated manner. Furthermore, the parameters of a monotonic fuzzy membership function can be set to resemble the crisp membership function, while the class weight can remain the same as in the crisp class rules.

2.3. Fuzzy model optimization

The fuzzy model is a generalization of the crisp model, having a larger number of parameters which can be tuned in order to create non-axis-parallel decision boundaries, which reflect the properties of a given data set more accurately. Thus, the fuzzy model $M^{\mathrm{f}}(x, \Theta^{\mathrm{f}}, t)$ is optimized with respect to its parameters Θ^{f} and/or t, using a training data set (D_{train}). A cost function (objective function) must be defined for this purpose. The optimization strategy can be either global or local [6]. Global optimization techniques attempt to find all local minima of an objective function $F(x)$ inside a bounded set $S \subset \mathbb{R}^n$, which are potentially global, while local optimization techniques start from an initial point and result to the respective local minimum. Global optimization has the advantage of (potentially) discovering the global minimum, however, may result to a solution that is not fully interpretable, i.e. the discovered solution (point) lies in a large distance from the initial point (generated from the crisp classifier). This may be handled setting the bounds of the search area S to limit the optimization in a relatively small area around the initial point; the points inside this area maintain the interpretation ability of the fuzzy model. Local optimization techniques do not suffer from this problem;

the resulted local minimum is in the area of the initial point and thus maintains its interpretation ability. However, other local minima in different areas from the initial point, potentially superior than the one discovered, cannot be found using local optimization. In addition, local optimization is significantly faster than global, even if we restrict global optimization into a small search area.

The fuzzy model has two parameter sets: Θ^{f}, which denotes the properties of the fuzzy membership function and t, which denotes the importance of each fuzzy class rule. Optimization can be performed subject to all parameters $\{\Theta^{\mathrm{f}}, t\}$, simultaneously or hybrid optimization schemes can be employed, where the optimization is applied in two stages; the fuzzy model is optimized subject to Θ^{f} while t is considered constant and vice versa.

3. A specific implementation

In order to apply the proposed methodology, several issues must be resolved, including:

- The origin of the initial crisp set of rules. As mentioned above, the implementation of the methodology presented here is data-driven.
- The definition of the technique which is used for the crisp set of rules mining from the initial annotated data set. There are two broad classes of methods extracting classification rules: (1) direct methods, which extract classification rules directly from the data [24], and (2) indirect methods, which extract classification rules from other classification models, such as decision trees [24].
- The fuzzy model's characteristics, such as the fuzzy membership function which are used for the fuzzification, T_{norm} and S_{norm} functions, which are used in the fuzzy inference engine, and the defuzzification function [30].
- The optimization strategy, the specific techniques (optimization algorithms) that will be employed and the objective function which will be minimized [6].

All the above are vital elements of the methodology and different choices and combinations among them result to different fuzzy models.

3.1. Crisp model creation based on decision tree induction

In our realization we used decision trees to extract the initial set of rules from the annotated data set, which is an indirect way for rule mining. The construction of the decision trees is implemented using the C4.5 inductive algorithm [21], which is an efficient and widely used decision tree-induction algorithm [20,22]. C4.5 generates a decision tree from the training data which minimizes the expected value of the number of tests for data classification. Each internal node of the tree corresponds to a feature a, while each outgoing branch corresponds to a feature test $g^{\mathrm{c}}(a, \theta^{\mathrm{c}}) = a\,op\,\theta^{\mathrm{c}}$. The leaf nodes represent the class to be assigned. The most important factor in the C4.5 algorithm is its ability to automatically select the feature, which is appropriate at each node. The feature of each node is selected in order to divide input samples efficiently. The information gain [21] is used as a measure of efficiency. After the induction of the decision tree, we apply a pruning method to reduce the tree's size and complexity. There exist two common methods for pruning: prepruning and postpruning. In our problem we followed the postpruning method, by replacing a subtree with a new leaf node whose class label is determined from the majority class of records associated with the subtree [21].

The produced tree can be easily transformed into a set of rules, as follows:

(a) One crisp rule $r^{\mathrm{c}}_{i,j}(x, \theta^{\mathrm{c}}_{i,j})$, having a crisp condition $Cond^{\mathrm{c}}_{i,j}$, is created for every leaf of the tree, by parsing the tree from the root node to that leaf. The feature tests encountered along the path form the conjuncts of the condition: $Cond^{\mathrm{c}}_{i,j}(x, \theta^{\mathrm{c}}_{i,j}) = g_{\mathrm{c}}(a_{i,j,1}, \theta^{\mathrm{c}}_{i,j,1}) \wedge g_{\mathrm{c}}(a_{i,j,2}, \theta^{\mathrm{c}}_{i,j,2}) \wedge \cdots \wedge g_{\mathrm{c}}(a_{i,j,K_{i,j}}, \theta^{\mathrm{c}}_{i,j,K_{i,j}})$, where $a_{i,j,1}, a_{i,j,2}, \ldots, a_{i,j,K_{i,j}}$ are the features encountered in the path and $\theta^{\mathrm{c}}_{i,j,1}, \theta^{\mathrm{c}}_{i,j,2}, \ldots, \theta^{\mathrm{c}}_{i,j,K_{i,j}}$ the respective parameters. The class label at the leaf node is assigned to the rule consequent: $r^{\mathrm{c}}_{i,j}(x, \theta^{\mathrm{c}}_{i,j}) : Cond^{\mathrm{c}}_{i,j}(x, \theta^{\mathrm{c}}_{i,j}) \to y_i$, $i = 1, \ldots, I$. Obviously, $a_{i,j,1}$ is the root feature and $\theta^{\mathrm{c}}_{i,j,1}$ is the root parameter, $K_{i,j}+1$ is the depth of the examined leaf and $n_l = \sum_{i=1}^{I} J_i$ is equal to the number of leafs in the decision tree (since one crisp rule is created for each leaf of the tree).

(b) A crisp class rule R^{c}_i is created for each class y_i, using all crisp rules having as consequent this class: $R^{\mathrm{c}}_i(x, \theta^{\mathrm{c}}_i) : (r^{\mathrm{c}}_{i,1} \vee r^{\mathrm{c}}_{i,2} \vee \cdots \vee r^{\mathrm{c}}_{i,J_i}) \to y_i$.

Thus, the crisp model M^c is defined as follows: $M^c(x, \Theta^c, t) = F^c(t_1 \cdot R_1^c, t_2 \cdot R_2^c, \ldots, t_I \cdot R_I^c), \Theta^c = \{\theta_1^c, \theta_2^c, \ldots, \theta_i^c\}$. The class weights t_i are initialized to one ($t_i = 1,\ i = 1, \ldots, I$) and their values which more accurately reflect the importance of each fuzzy class rule are obtained in the optimization stage.

3.2. Development of a fuzzy model

The crisp model (M^c) is transformed into a fuzzy model (M^f) as follows:

- The sigmoid function is used as fuzzy membership function instead of the crisp membership function. The sigmoid function is defined as

$$g^f(a, \theta^f) = \frac{1}{1 + e^{\theta^{1,f}(a-\theta^{2,f})}}, \tag{3}$$

  where θ^f is a vector containing all parameters used in the sigmoid function, $\theta^f = \{\theta^{1,f}, \theta^{2,f}\}$.
- The binary AND and OR operators are replaced with the T_{norm} and S_{norm} functions, defined as min and max operators [30], respectively.
- The F^c function is replaced with a defuzzification function F^f, selected as the maximum defuzzifier [30].

According to the above, each fuzzy rule $r_{i,j}^f(x, \theta_{i,j}^f) \to y_i$ is defined as $r_{i,j}^f(x, \theta_{i,j}^f) = Cond_{i,j}^f(x, \theta_{i,j}^f)$, $Cond_{i,j}^f = \min(g^f(a_{i,j,1}, \theta_{i,j,1}^f), g^f(a_{i,j,2}, \theta_{i,j,2}^f), \ldots, g^f(a_{i,j,K_{i,j}}, \theta_{i,j,K_{i,j}}^f))$. The fuzzy class rules ($R_i^f(x, \theta_i^f) \to y_i, i = 1, \ldots, I$) are defined as

$$\begin{aligned} R_i^f(x, \theta_i^f) &= \max(r_{i,1}^f, r_{i,2}^f, \ldots, r_{i,J_i}^f) \\ &= \max\begin{pmatrix} \min(g^f(a_{i,1,1}, \theta_{i,1,1}^f), g^f(a_{i,1,2}, \theta_{i,1,2}^f), \ldots, g^f(a_{i,1,K_{i,j}}, \theta_{i,1,K_{i,j}}^f)), \\ \min(g^f(a_{i,2,1}, \theta_{i,2,1}^f), g^f(a_{i,2,2}, \theta_{i,2,2}^f), \ldots, g^f(a_{i,2,K_{i,j}}, \theta_{i,2,K_{i,j}}^f)), \\ \ldots, \\ \min(g^f(a_{i,J_i,1}, \theta_{i,J_i,1}^f), g^f(a_{i,J_i,2}, \theta_{i,J_i,2}^f), \ldots, g^f(a_{i,J_i,K_{i,j}}, \theta_{i,J_i,K_{i,j}}^f)) \end{pmatrix}. \end{aligned} \tag{4}$$

Finally, M^f is defined as $M^f(x, \Theta^f, t) = \max(t_1 \cdot R_1^f, t_2 \cdot R_2^f, \ldots, t_I \cdot R_I^f)$. Thus:

$$\begin{aligned} &M^f(x, \Theta^f, t) \\ &\quad = \max(t_1 \cdot R_1^f, t_2 \cdot R_2^f, \ldots, t_I \cdot R_I^f) \\ &\quad = \max\begin{pmatrix} t_1 \cdot \max\begin{pmatrix} \min(g^f(a_{1,1,1}, \theta_{1,1,1}^f), g^f(a_{1,1,2}, \theta_{1,1,2}^f), \ldots, g^f(a_{1,1,K_{1,1}}, \theta_{1,1,K_{1,1}}^f)), \\ \min(g^f(a_{1,2,1}, \theta_{1,2,1}^f), g^f(a_{1,2,2}, \theta_{1,2,2}^f), \ldots, g^f(a_{1,2,K_{1,2}}, \theta_{1,2,K_{1,2}}^f)), \\ \ldots, \\ \min(g^f(a_{1,J_1,1}, \theta_{1,J_1,1}^f), g^f(a_{1,J_1,2}, \theta_{1,J_1,2}^f), \ldots, g^f(a_{1,J_1,K_{1,J_1}}, \theta_{1,J_1,K_{1,J_1}}^f)) \end{pmatrix}, \\ t_2 \cdot \max\begin{pmatrix} \min(g^f(a_{2,1,1}, \theta_{2,1,1}^f), g^f(a_{2,1,2}, \theta_{2,1,2}^f), \ldots, g^f(a_{2,1,K_{2,1}}, \theta_{2,1,K_{2,1}}^f)), \\ \min(g^f(a_{2,2,1}, \theta_{2,2,1}^f), g^f(a_{2,2,2}, \theta_{2,2,2}^f), \ldots, g^f(a_{2,2,K_{2,2}}, \theta_{2,2,K_{2,2}}^f)), \\ \ldots, \\ \min(g^f(a_{2,J_2,1}, \theta_{2,J_2,1}^f), g^f(a_{2,J_2,2}, \theta_{2,J_2,2}^f), \ldots, g^f(a_{2,J_2,K_{2,J_2}}, \theta_{2,J_2,K_{2,J_2}}^f)) \end{pmatrix}, \\ \ldots, \\ t_I \cdot \max\begin{pmatrix} \min(g^f(a_{I,1,1}, \theta_{I,1,1}^f), g^f(a_{I,1,2}, \theta_{I,1,2}^f), \ldots, g^f(a_{I,1,K_{I,1}}, \theta_{I,1,K_{I,1}}^f)), \\ \min(g^f(a_{I,2,1}, \theta_{I,2,1}^f), g^f(a_{I,2,2}, \theta_{I,2,2}^f), \ldots, g^f(a_{I,2,K_{I,2}}, \theta_{I,2,K_{I,2}}^f)), \\ \ldots, \\ \min(g^f(a_{I,J_I,1}, \theta_{I,J_I,1}^f), g^f(a_{I,J_I,2}, \theta_{I,J_I,2}^f), \ldots, g^f(a_{I,J_I,K_{I,J_I}}, \theta_{I,J_I,K_{I,J_I}}^f)) \end{pmatrix} \end{pmatrix} \\ &= \max_{i=1}^{I}\left(t_i \cdot \max_{j=1}^{J_i}\left(\min_{k=1}^{K_{i,j}}(g^f(a_{i,j,k}, \theta_{i,j,k}^f))\right)\right) = \max_{i=1}^{I}\left(\max_{j=1}^{J_i}\left(t_i \cdot \min_{k=1}^{K_{i,j}}\left(1 + e^{\theta_{i,j,k}^{1,f}(a_{i,j,k}-\theta_{i,j,k}^{2,f})}\right)^{-1}\right)\right) \\ &\Rightarrow M^f(x, \Theta^f, t) = \max_{i=1,j=1}^{I,J_i}\left(\min_{k=1}^{K_{i,j}}(t_i \cdot (1 + e^{\theta_{i,j,k}^{1,f}(a_{i,j,k}-\theta_{i,j,k}^{2,f})})^{-1})\right), \end{aligned} \tag{5}$$

since $t_i \geqslant 0$, while $\Theta^{\mathrm{f}} = \{\theta_1^{\mathrm{f}}, \theta_2^{\mathrm{f}}, \ldots, \theta_I^{\mathrm{f}}\}$. Eq. (5) denotes the implicit input–output formula of the fuzzy model. The fuzzy model (M^{f}) is a generalization of the respective crisp (M^{c}) model, since:

if (i) $t_i = 1$,
(ii) $\theta_{i,j,k}^{1,\mathrm{f}} \to +\infty$, if the crisp membership function is decreasing or
$\theta_{i,j,k}^{1,\mathrm{f}} \to -\infty$, if the crisp membership function is increasing,
(iii) $\theta_{i,j,k}^{2,\mathrm{f}} = \theta_{i,j,k}^{\mathrm{c}}$
then $M^{\mathrm{f}} \to M^{\mathrm{c}}$.

3.3. Fuzzy model parameter optimization

The fuzzy model $M^{\mathrm{f}}(x, \Theta^{\mathrm{f}}, t)$ is optimized with respect to its parameters Θ^{f} and t, using a training data set (D_{train}). For this purpose, a cost function is used, defined as $F(\Theta, D_{\mathrm{train}}) = \mathrm{trace}(X)/|D_{\mathrm{train}}|$, where X is the confusion matrix, defined as $X_{M^{\mathrm{f}}(x,\Theta^{\mathrm{f}},t),y} =$ *number of patterns in class y classified to class* $M^{\mathrm{f}}(x, \Theta^{\mathrm{f}}, t)$, and $|D_{\mathrm{train}}|$ is the size (number of patterns) included in the D_{train}. Two optimization techniques have been employed, the Direct method [19] for global optimization, and the Nelder–Mead simplex search method [16] for local optimization.

Direct [19] is a global optimization algorithm which can be used in problems with bound constraints and a real-valued objective function. It requires no knowledge of the objective function's gradient. The algorithm sample points in the domain and uses this information to decide where to search next. Therefore, it can be applied in cases where the objective function is considered as a "black box'' function. The Nelder–Mead simplex search method [16] is an unconstrained nonlinear local optimization technique, which attempts to find a minimum of a scalar function of several variables, starting from an initial estimate (initial point). Initially, a simplex $\Theta \in \mathbb{R}^n$ in the n-dimensional space is defined, which is characterized by the $n + 1$ distinct vectors being its vertices. At each step of the search, a new point in or near the current simplex is generated. The function value at the new point is compared with the function's values at the vertices of the simplex and, usually, one of the vertices is replaced by the new point, giving a new simplex. This step is repeated until the diameter of the simplex is less than the specified tolerance. The Nelder–Mead simplex search method does not use numerical or analytical computation of the gradient.

We have tested our methodology, employing four different optimization strategies:

1. Optimization of the Θ^{f} parameters using the Direct method. The bounds for the parameters were defined as: (i) the upper and the lower bound for the $\theta_{i,j,k}^{1,\mathrm{f}}$ parameters are set to 0 and 10, respectively, and (ii) the upper and the lower bound for the $\theta_{i,j,k}^{2,\mathrm{f}}$ parameters are set to $0.75 \cdot \theta_{i,j,k}^{\mathrm{c}}$ and $1.25 \cdot \theta_{i,j,k}^{\mathrm{c}}$ ($\theta_{i,j,k}^{\mathrm{c}}$ are defined from the decision tree), respectively. Fuzzy class rule weights (parameters t) were considered as constants, and set to 1 ($t_i = 1$).

(1) Optimization of the Θ^{f} parameters using the Nelder–Mead simplex search method. The initial point was defined setting: (i) $\theta_{i,j,k}^{2,\mathrm{f}} = \theta_{i,j,k}^{\mathrm{c}}$ ($\theta_{i,j,k}^{\mathrm{c}}$ are defined from the decision tree) and (ii) $\theta_{i,j,k}^{1,\mathrm{f}} \sim \mathrm{N}(5, 1)$, if the crisp membership function is decreasing or $\theta_{i,j,k}^{1,\mathrm{f}} \sim -\mathrm{N}(5, 1)$, if the crisp membership function is increasing. Again, the fuzzy class rule weights (parameters t) were considered as constants, and set to 1 ($t_i = 1$).

(2) Hybrid two-stage optimization strategy (hybrid 1):
   (a) Set $t_i = 1$ and optimize M^{f} with respect to Θ^{f}, using the Direct method (as described above). This results to an optimal set of Θ^{f} parameters, $\Theta^{\mathrm{f}*}$.
   (b) Set $\Theta^{\mathrm{f}} = \Theta^{\mathrm{f}*}$ and optimize M^{f} with respect to t, using the Nelder–Mead method. The fuzzy class rule weights are initialized as $t_i \sim \mathrm{U}(0.95, 1.05)$.

(3) Hybrid two-stage optimization strategy (hybrid 2):
   (a) Set $t_i = 1$ and optimize M^{f} with respect to Θ^{f}, using the Nelder–Mead method. This results to an optimal set of Θ^{f} parameters, $\Theta^{\mathrm{f}*}$.
   (b) Set $\Theta^{\mathrm{f}} = \Theta^{\mathrm{f}*}$ and optimize M^{f} with respect to t, using the Nelder–Mead method. Again, the fuzzy class rules weights are initialized as $t_i \sim \mathrm{U}(0.95, 1.05)$.

A working example of the above realization is presented in the Appendix.

4. Results

Extensive evaluation was performed for the described realization of the proposed methodology. Results are presented in terms of classification accuracy, using all four optimization strategies.

In order to evaluate the proposed methodology several well-known data sets obtained from the UCI machine learning repository [3] were employed. Table 1 presents all data sets which were employed, along with the number of samples included, the number of attributes used in each data set and the number of classes. These data sets were selected because of the small number of missing values; none or very few values are missing in each data set. Missing values are replaced by the average of the column (attribute).

The 10 fold stratified cross-validation method [24], was used for the evaluation. The procedure was applied to each fold, generating 10 different decision trees and, subsequently, 10 different crisp and fuzzy models. We have compared the results obtained by our methodology with the results obtained by six widely used classification methodologies: decision trees, naïve Bayes classifier, k nearest-neighbors (k-NN), multilayer perceptron (MLP) neural networks, radial basis functions (RBF) neural networks, and support vector machines (SVM). We have selected the parameters for the classifiers as follows [32]:

(i) Decision trees were implemented using the C4.5 algorithm. Postpruning was employed, using the pessimistic error rate-based method (subtree replacement). The confidence factor for pruning was set to 0.25 and the minimum number of instances in a leaf was 2.
(ii) The naïve Bayes classifier was implemented by modeling numerical values using normal distributions.
(iii) k-NN was implemented based on the normalized Euclidean distance. Several experiments were carried out for different values of k and the best average results for all data sets are obtained for $k = 3$.
(iv) MLP architecture was implemented with the number of hidden layers set to 1 and the number of hidden neurons set as $(\#features + \#classes)/2$, while the learning rate was set to 0.3, the momentum to 0.2 and the number of epochs to 500.
(v) RBF were implemented using the K-Means clustering algorithm to provide the basis functions and then by learning a logistic regression function.
(vi) SVM were implemented using RBF kernel. Grid search was employed to identify the values for the c and gamma parameters, which provide the best results for each data set.

Table 2 presents the total accuracy (average accuracy of the 10 fold and standard deviation) for all 16 data sets, with all six classifiers and the proposed methodology, using all four different optimization strategies (global, local, hybrid 1, hybrid 2). In addition, overall results are presented, for each classifier, in terms of average accuracy for all data sets

Table 1
Description of the data sets used for the evaluation of the proposed methodology [3]

Dataset	Samples	Attributes	Classes
1985 Auto Imports Database (autos)	205	25	6
Wisconsin breast cancer (breast_c)	699	9	2
Credit approval (credit_a)	690	15	2
German credit data (credit_g)	1000	20	2
Protein localization sites (ecoli)	332	7	6
Glass identification database (glass)	214	9	6
Cleveland heart disease (heart_c)	303	13	2
Heart disease (heart_statlog)	270	13	2
Johns Hopkins University Ionosphere database (ionosphere)	351	34	2
Iris plants database (iris)	150	4	3
BUPA liver disorders (liver_d)	345	6	2
Pima Indian diabetes (pima_d)	768	8	2
Image segmentation data (segments)	2310	19	7
Sonar, mines vs. rocks (sonar)	208	60	2
Vehicle silhouettes (vehicle)	846	18	4
Wine recognition data (wine)	178	13	3

Table 2
Comparison of the results obtained from six well-known classifiers and the proposed methodology, employing global, local, hybrid 1 and hybrid 2 optimization strategies

Datasets	Global	Local	Hybrid 1	Hybrid 2	Decision tree	3-NN	Naïve Bayes	MLP	RBF	SVM
Autos	80.93 ± 7.3	**84.36** ± 6.54	83.98 ± 6.5	84.36 ± 6.54	83.88 ± 6.21	70.14 ± 10.52	57.14 ± 13.71	77.93 ± 8.35	60.55 ± 8.86	69.71 ± 7.88
breast_c	95.14 ± 2.15	94.99 ± 2.16	95.71 ± 2.43	95 ± 2.05	94.99 ± 2.05	**96.85** ± 1.89	95.99 ± 1.63	95.42 ± 3.22	95.56 ± 2.38	**96.85** ± 1.64
credit_a	**85.94** ± 3.13	**85.94** ± 3.13	**85.94** ± 3.13	**85.94** ± 3.13	85.8 ± 3.4	85.36 ± 3.38	77.68 ± 3.15	83.04 ± 4.98	79.71 ± 3.35	84.93 ± 4.49
credit_g	71.8 ± 2.39	71.2 ± 2.7	72.5 ± 2.76	72.1 ± 2.18	71.2 ± 2.7	73.3 ± 3.43	**75.4** ± 4.3	71.6 ± 3.03	74 ± 5.14	70.20 ± 0.63
ecoli	84.35 ± 6.15	83.74 ± 6.85	84.95 ± 6.2	84.05 ± 6.52	84.05 ± 6.97	85.85 ± 5.1	86.48 ± 5.78	86.76 ± 6.36	83.73 ± 4.11	**88.26** ± 4.78
glass	70.11 ± 3.73	69.18 ± 4.83	70.39 ± 3.45	70.07 ± 7.5	69.18 ± 4.33	**71.95** ± 5.42	48.59 ± 6.55	66.82 ± 9.55	64.46 ± 10.85	69.59 ± 5.77
heart_c	75.28 ± 10.28	75.28 ± 10.28	76.27 ± 10.46	75.61 ± 10.73	75.28 ± 10.28	81.11 ± 7.63	82.82 ± 6.58	80.83 ± 6.12	**84.12** ± 6.96	80.85 ± 6.83
heart_statlog	81.11 ± 7.29	78.14 ± 5.91	81.11 ± 7.29	79.26 ± 7.24	78.52 ± 6.94	78.89 ± 5.53	83.7 ± 7.03	78.15 ± 7.08	**84.07** ± 6.54	80.00 ± 6.58
ionosphere	88.61 ± 4.83	88.61 ± 4.83	89.18 ± 4.97	89.74 ± 5.75	88.61 ± 4.83	86.6 ± 4.69	82.62 ± 5.47	91.17 ± 2.81	92.62 ± 5.66	**95.17** ± 3.28
iris	94 ± 6.63	94 ± 6.63	96 ± 5.62	95.33 ± 5.49	94 ± 6.63	95.33 ± 5.49	96 ± 4.66	**97.33** ± 3.44	95.33 ± 4.5	96.67 ± 4.71
liver_d	66.91 ± 6.69	66.65 ± 5.08	68.57 ± 6.21	67.22 ± 6.52	66.65 ± 5.08	61.73 ± 5.88	55.39 ± 8.86	**71.55** ± 7.38	64.35 ± 6.86	71.24 ± 9.19
pima_d	76.29 ± 5.12	76.03 ± 5.25	**77.34** ± 5.13	76.68 ± 4.58	76.68 ± 5.17	72.65 ± 5.14	76.31 ± 5.52	75.4 ± 4.66	75.4 ± 4.36	76.17 ± 5.19
segments	95.71 ± 0.92	96.97 ± 0.96	95.58 ± 1.02	95.8 ± 0.73	**96.97** ± 0.96	96.02 ± 0.64	80.22 ± 1.89	96.06 ± 1.28	87.19 ± 1.87	96.80 ± 1.00
sonar	72.62 ± 8.02	72.62 ± 8.02	75.5 ± 4.62	75 ± 6.64	72.62 ± 8.02	86.02 ± 7.07	67.88 ± 9.29	82.29 ± 10.7	72.12 ± 7	**88.50** ± 7.48
vehicle	69.40 ± 5.22	74.95 ± 3.22	73.53 ± 3.96	72.59 ± 4.74	73.53 ± 4.06	71.52 ± 6.04	44.8 ± 4.72	81.7 ± 3.82	66.68 ± 6.33	**84.99** ± 2.76
wine	93.27 ± 3.49	92.68 ± 2.77	92.19 ± 4.67	92.68 ± 2.77	93.27 ± 3.61	94.97 ± 4.11	96.63 ± 5.38	97.19 ± 3.96	98.3 ± 2.74	**97.22** ± 2.93
Overall (mean)	81.34 ± 5.21	81.58 ± 4.95	82.42 ± 4.90	81.96 ± 5.19	81.58 ± 5.08	81.77 ± 5.12	75.48 ± 5.91	83.33 ± 5.42	79.89 ± 5.47	**84.2** ± 4.7

Highlighted (bold) results denote the best result for each data set.

and average standard deviation. The proposed methodology presents overall accuracy 81.34%, 81.56%, 82.42% and 81.96%, when global, local, hybrid 1 and hybrid 2 optimization strategies are employed, respectively.

Comparing the results of the proposed methodology when global and local optimization strategies are used, local optimization reports higher overall accuracy; however, global optimization has better results in half of the data sets, worse in three and in five data sets the same accuracy with local optimization is reported. Concerning the hybrid optimization strategies, hybrid 1 reports higher overall accuracy and also reports higher results in 11 of the 16 data sets and the same in one. The use of the hybrid 1 optimization technique, which is an extension of the global one, improves the respective results in 12 cases, while in two cases the results remain the same. In the same context, the use of the hybrid 2 optimization technique, which is an extension of the local one, improves the results in 10 cases, while in 4 cases the results remain the same. In both cases, for the two data sets, the additional optimization step of the hybrid approaches results in a decrease of the classification accuracy.

As it is described above, the proposed methodology using the global or local optimization techniques is expected to improve the classification ability of the respective decision tree, due to the fuzzification step. Furthermore, the weight introduction and optimization (hybrid approaches) is expected to further increase the classification ability, since it provides more flexibility into the fuzzy model. Fig. 2 presents the accuracy variation for two cases: (i) decision tree to global optimization and then to hybrid 1 optimization and (ii) decision tree to local optimization and then to hybrid 2 optimization. In both cases, the experimental results follow the expected behavior; in the first case (decision tree – global – hybrid 1), this trend is observed in 12 classification problems, while in the second case (decision tree – local – hybrid 2), in 9. It should be mentioned that in 14 of the total 16 classification data sets, the proposed methodology increases the classification accuracy of the respective decision tree, while in the remaining two the classification accuracy remains the same.

5. Discussion

In this paper we propose a generic methodology for the automated generation of fuzzy models, which consists of three stages. Initially, a crisp model is generated, then it is transformed to a fuzzy model and, finally, the parameters entering the fuzzy model are optimized. A specific realization of the above methodology, consisting of: (i) generation of a crisp model from a decision tree, induced from the data, (ii) transformation of the crisp model into a fuzzy one, using the sigmoid function, as fuzzy membership function, the min and max operators for T_{norm} and S_{norm} functions, respectively, and the maximum defuzzifier and (iii) optimization of the fuzzy model's parameters, is presented. The optimization stage has been addressed using global, local and hybrid optimization. This realization has been extensively evaluated, using 16 benchmark data sets from the UCI machine learning repository. The obtained results indicate high classification ability, comparable or better than well-known classification schemes.

The proposed methodology can integrate any rule-mining technique to generate the crisp model. This is a major advantage, compared to several approaches proposed in the literature, which are based on a specific tree-induction algorithm or knowledge-based approaches. Approaches based on a specific tree-induction algorithm [34,12,13,31] are limited, due to the disadvantages of the specific technique employed, such as the ID3 tree-induction algorithm [17,7]. In addition, they cannot integrate new advances in the area, since they are adapted only to a specific technique. On the other hand, approaches based on knowledge [28,27] are limited to domains where expert knowledge is available. The proposed methodology has the ability to integrate knowledge-based rules or a mixture of rules obtained by experts and rules obtained using rule mining. In general, all stages of the methodology are generic (crisp model generation, fuzzy modeling, optimization) and can integrate new advances and techniques in their respective fields. Also, the methodology is realized in a fully automated manner, since the transformation of the crisp model to the fuzzy one is straightforward.

The proposed realization, which falls in the third of the categories which are mentioned in the Introduction (i.e. induction of a crisp decision tree, extraction of a set of rules from it and fuzzification of these rules), presents several similarities with the approaches from the other two categories (i.e. induction of a crisp decision tree from the data and then its fuzzification, resulting to a fuzzy decision tree or induction of fuzzy decision tree, integrating fuzzy techniques during the tree construction). However, an important difference is that in a fuzzy decision tree, each node receives a single value for each of its parameters, thus having the same decision making functionality for all child-leaves. In the case of rule extraction from the tree, each node is part of more than one rule (i.e. it is included as a conjunct in all the rules corresponding to its child-leaves) and receives different parameter values for each one of them. This increases the optimization time required for the fuzzy rules compared to the fuzzy decision trees due to the larger number of

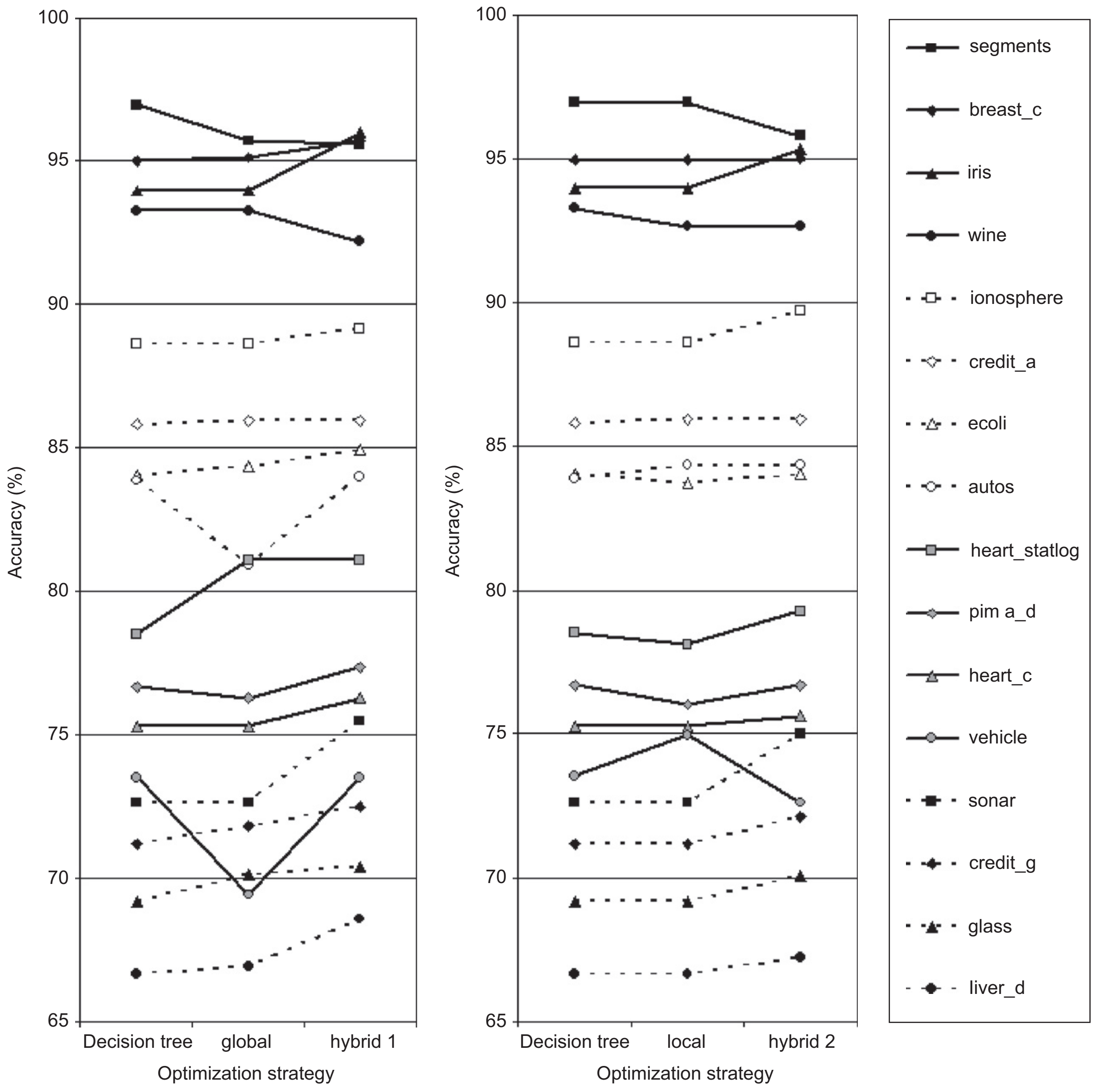

Fig. 2. Graphic representation of the accuracy variation: (a) use of global optimization and then hybrid 1 optimization strategy and (b) use of local optimization and then hybrid 2 optimization strategy, for all 16 data sets.

parameters which are introduced. However, it also bestows a major advantage on the decision making procedure of the fuzzy rules compared to the fuzzy decision trees since the larger number of parameters allows the fuzzy rules to be more flexible and thus more adaptable to a specific data set. In addition, the complexity of the decision making process remains the same (the number of nodes that have to be parsed to reach a decision is the same with the number of conjuncts of the respective rule).

In the fuzzy model two sets of parameters are optimized: Θ^f, which includes all parameters included in the sigmoid functions and t, which are the class rule's weights. As previously mentioned, two known approaches are presented: (i) use of local weights and (ii) use of global weights. The introduction of the class weights (t parameters) is a novel approach, and constitutes a major advantage of our methodology, since the final fuzzy model, besides the adaptation

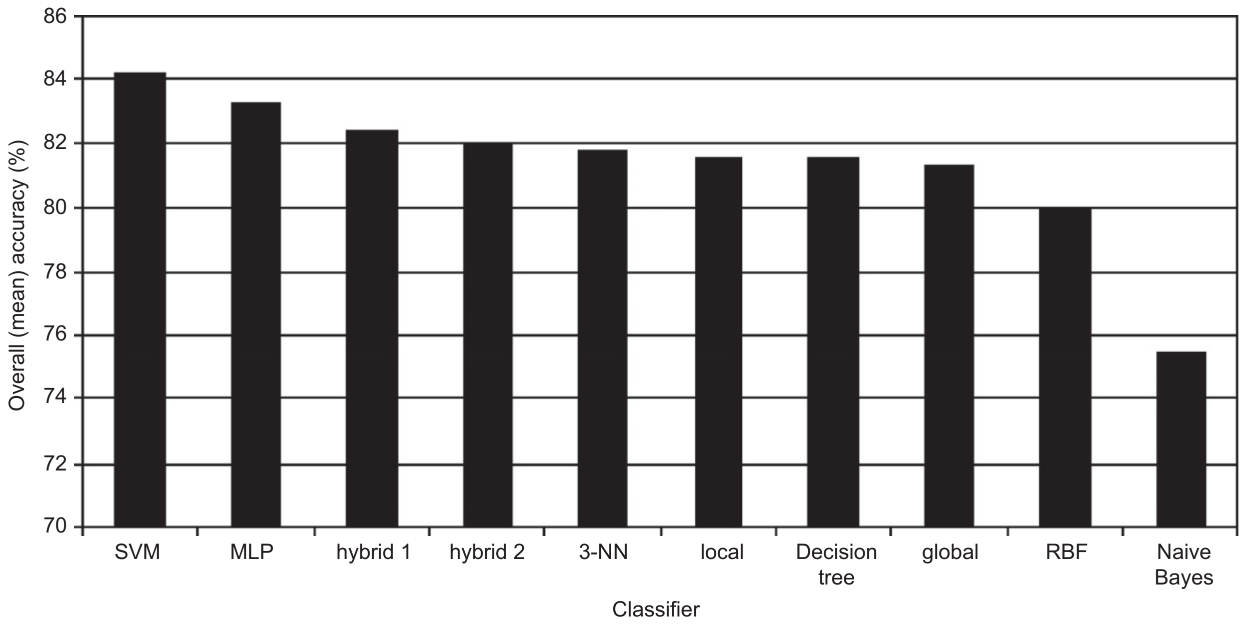

Fig. 3. Graphic representation of the overall accuracy of all classifiers.

concerning the membership functions, it has also the ability to identify the individual importance of each class and thus, to more accurately reflect the underlying properties of the application domain. Also, the class weights are advantageous compared to local or global weights, since local or global weights are not adapted to the classes. However, the integration of all three types of weights (local, global and class) would be of great interest; this will be addressed in a future communication.

Concerning the evaluation procedure, a large number of benchmark data sets has been employed, with different number of instances, features and classes, having different classification complexity. Also, all evaluation experiments were conducted using 10-fold stratified cross-validation, which is considered as the most reliable approach for evaluation [15]. These features ensure that the evaluation is adequate to demonstrate the advantages/disadvantages of our methodology and fully exploit its potential. This is an advantage, compared to several similar approaches proposed in the literature which: (i) mainly focus on a specific domain of application [28,8,27], (ii) employ one or a small number of data sets [14,5,7,34,12,31] for evaluation or (iii) have been evaluated using different evaluation strategies [23,1,8,27].

A comparative study of our methodology with six widely used classification methodologies has been conducted. Fig. 3 presents the overall accuracy (mean accuracy of all data sets) of the 10 different classification schemes: the proposed methodology (with the four different optimization strategies) and the six classifiers of the comparative study. The classifiers are ordered by the overall accuracy. SVMs presented the highest accuracy (84.2%), followed by MLPs (83.33%), while the proposed methodology with hybrid 1 optimization strategy (82.42%) and hybrid 2 optimization strategy (81.96%) follow. However, both SVMs and MLPs are considered as black box approaches, lacking interpretation ability. In addition, SVMs is the only classifier where grid search was employed, thus, the best values of the parameters were identified for each data set; this feature was neither used in any of the comparative classifiers nor in our method. Also, several experiments were conducted using the k-NN classifier, to obtain the best value for k; however, this was calculated based on the average accuracy over all data sets and not by calculating the best k value for each data set. As it is shown in Fig. 3, the overall accuracy of decision trees is higher than the one obtained from the proposed methodology using global optimization. However, a direct comparison for each data set shows that the proposed methodology using global optimization outperforms decision trees in seven data sets, while for five data sets both present the same accuracy; decision trees report better results in four cases.

Table 3 presents a comparison of the results obtained by similar approaches presented in the literature. The nine data sets included (breast_c, credit_g, glass, heart_c, ionosphere, iris, pima_d, sonar and wine) are those which are reported in at least two of these research attempts while the studies of Suarez et al. [23], Abonyi et al. [1], Crockett et al.

Table 3
Comparison of the results obtained by six well-known classifiers and the proposed methodology, employing global, local, hybrid 1 and hybrid 2 optimization strategies

Datasets	Previous works					Proposed methodology			
	Suarez et al. [28]	Suarez et al. [28]	Abonyi et al. [31]	Crockett et al. [17]	Olaru et al. [14]	global	local	hybrid 1	hybrid 2
breast_c	95.8	95.9	**96.82**	95.58		95.14	94.99	95.71	95
credit_g	73.4	**73.8**				71.80	71.20	72.5	72.1
glass			66.03		**70.91**	70.11	69.18	70.39	70.07
heart_c	77.6	74		**77.7**	74.41	75.28	75.28	76.27	75.61
ionosphere			86.47		89.64	88.61	88.61	89.18	**89.74**
iris			**96.11**		95	94.00	94.00	96	95.33
pima_d	74.8	74.3	73.05	**78.5**	74.43	76.29	76.03	77.34	76.68
sonar	74.9	70.9			72.56	72.62	72.62	**75.5**	75
wine			91.22		**96.47**	93.27	92.68	92.19	92.68
	79.30	77.78				78.23	78.02	79.46	78.88
Overall			84.95			86.24	85.92	86.8	86.58
(mean)				83.93		82.24	82.10	83.11	82.43
					81.92	81.45	81.20	82.41	82.16

Highlighted (bold) results denote the best result for each data set.

[7] and Olaru et al. [17] have presented results for at least three data sets, also used in the evaluation of the proposed methodology. Overall accuracy results (mean values) are also presented in Table 3. Each line presents the mean accuracy for the data sets used in each work, i.e. in the first line of the overall section the mean accuracy corresponds only to the data sets employed by Suarez et al., in the second the mean accuracy corresponds only to the data sets employed by Abonyi et al., etc.

Suarez et al. [23] used five common data sets for the evaluation of two different architectures, reporting an average 79.3% accuracy for the first and 77.78% accuracy for the second, respectively. The second architecture presents lower average results compared to all four different optimization strategies, while the first is outperformed only when hybrid 1 optimization strategy is employed. It should be mentioned that the evaluation performed in [23] is based on 10 different randomly selected training-test sets while in the proposed work, 10-fold stratified cross-validation is used, which is considered more reliable. The work of Abonyi et al. [1] uses six common data sets, reporting average accuracy 84.95%, which is lower than the accuracy reported from all four different optimization strategies of the proposed method. The evaluation performed in [1] is based on 5-fold cross-validation. In [1] a model simplification stage is included, thus the generated fuzzy models are simpler (i.e. have less fuzzy rules) than the ones generated in this study. The approach proposed by Crockett et al. [7] presents better overall results compared to all four optimization strategies, but a relatively small number of data sets has been used (five in total, three are common). Finally, the method presented by Olaru et al. [17] reports higher average results compared to the global and local optimization strategies, while both hybrid optimization strategies present better average results.

Future work will focus on the introduction of more advanced techniques for the transformation of the crisp model to the respective fuzzy. Furthermore, in this work no effort was made to reduce the complexity of the generated fuzzy model, which can be handled during the decision tree pruning or by pruning or merging the fuzzy rules. Finally, the generalization ability of the methodology could be more reliable if validation subsets are employed in the training procedure as stopping criteria.

6. Conclusions

In this paper we present a generic methodology for automated fuzzy model generation along with a specific data-driven realization. The methodology is designed so as the generated fuzzy models to be more flexible in their decision boundaries than fuzzy models generated from previously reported methods. In addition, it can integrate state-of-the-art advances in any of the stages (rule mining, fuzzy modeling), while class weights, which is a novel approach, is shown to increase the classification ability of the generated fuzzy models. Sixteen benchmark data sets have been employed to evaluate the realization of the methodology and high accuracy results have been reported, improving the accuracy of the initial crisp models, and being comparable or better than the results of other widely used classification schemes and other similar methodologies.

Acknowledgments

This research is part funded by the program "Heraklitos'' of the Operational Program for Education and Initial Vocational Training of the Hellenic Ministry of Education under the 3rd Community Support Framework and the European Social Fund.

Appendix A

In this appendix we provide a working example of the realization of the methodology. The application is based on the iris data set [3] and more specifically on the 4th fold of the 10 folds used in the evaluation. The iris data set has four features ($n_f = 4$): $x = \{sepallength, sepalwidth, petallength, petalwidth\}$ and three classes ($n_y = 3$), being $\{setosa, versicolor, virginica\}$.

A.1. Crisp model creation based on decision tree induction

Initially, a decision tree is created:

$petalwidth \leqslant 0.4 : setosa$

$petalwidth > 0.4$

$|\ petalwidth \leqslant 1.7$

$|\ |\ petallength \leqslant 4.9 : versicolor$

$|\ |\ petallength > 4.9 : virginica$

$|\ petalwidth > 1.7 : virginica$

and by parsing it the following set of rules is created:

$$\begin{aligned}
&\textit{if } (petalwidth \leqslant 0.4) \textit{ then } c = setosa,\\
&\textit{if } (petalwidth > 0.4 \wedge petalwidth \leqslant 1.7 \wedge petallength \leqslant 4.9) \textit{ then } c = versicolor,\\
&\textit{if } (petalwidth > 0.4 \wedge petalwidth \leqslant 1.7 \wedge petallength > 4.9) \textit{ then } c = virginica,\\
&\textit{if } (petalwidth > 0.4 \wedge petalwidth > 1.7) \textit{ then } c = virginica.
\end{aligned}$$

Thus, the following conditions are obtained:

$$\begin{aligned}
&Cond^{c}_{setosa,1}(x, \theta^{c}_{setosa,1}) = g^{c}(\textit{petalwidth}, 0.4),\\
&Cond^{c}_{versicolor,1}(x, \theta^{c}_{versicolor,1}) = g^{c}(\textit{petalwidth}, 0.4) \wedge g^{c}(\textit{petalwidth}, 1.7) \wedge g^{c}(\textit{petallength}, 4.9),\\
&Cond^{c}_{virginica,1}(x, \theta^{c}_{virginica,1}) = g^{c}(\textit{petalwidth}, 0.4) \wedge g^{c}(\textit{petalwidth}, 1.7) \wedge g^{c}(\textit{petallength}, 4.9),\\
&Cond^{c}_{virginica,2}(x, \theta^{c}_{virginica,2}) = g^{c}(\textit{petalwidth}, 0.4) \wedge g^{c}(\textit{petalwidth}, 1.7).
\end{aligned}$$

The crisp membership function g^{c} is defined as increasing or decreasing, according to the crisp operator ($\leqslant$ or $>$). Thus, $\theta^{c}_{setosa,1} = \{0.4\}$, $\theta^{c}_{versicolor,1} = \{0.4, 1.7, 4.9\}$, $\theta^{c}_{virginica,1} = \{0.4, 1.7, 4.9\}$ and $\theta^{c}_{virginica,2} = \{0.4, 1.7\}$.

The respective crisp class rules are:

$$\begin{aligned}
&R^{c}_{setosa}(x, \theta^{c}_{setosa}) : (g^{c}(\textit{petalwidth}, 0.4)) \to setosa,\\
&R^{c}_{versicolor}(x, \theta^{c}_{versicolor}) :\\
&\quad (g^{c}(petalwidth, 0.4) \wedge g^{c}(petalwidth, 1.7) \wedge g^{c}(petallength, 4.9)) \to versicolor,
\end{aligned}$$

$$R^{c}_{virginica}(x, \theta^{c}_{virginica}) :$$
$$\begin{pmatrix} (g^{c}(petalwidth, 0.4) \wedge g^{c}(petalwidth, 1.7) \wedge g^{c}(petallength, 4.9)) \vee \\ (g^{c}(petalwidth, 0.4) \wedge g^{c}(petalwidth, 1.7)) \end{pmatrix} \rightarrow virginica.$$

Thus,

$$\theta^{c}_{setosa} = \{\theta^{c}_{setosa,1}\} = \{0.4\}, \quad \theta^{c}_{versicolor} = \{\theta^{c}_{versicolor,1}\} = \{0.4, 1.7, 4.9\}$$

and

$$\theta^{c}_{virginica} = \{\theta^{c}_{virginica,1}, \theta^{c}_{virginica,2}\} = \{0.4, 1.7, 4.9, 0.4, 1.7\}.$$

A.3. Development of a fuzzy model

The following fuzzy conditions are obtained:

$$Cond^{f}_{setosa,1}(x, \theta^{f}_{setosa,1}) = \min(g^{f}(petalwidth, \theta^{1,f}_{setosa,1,1}, 0.4)),$$

$$Cond^{f}_{versicolor,1}(x, \theta^{f}_{versicolor,1}) = \min \begin{pmatrix} g^{f}(petalwidth, \theta^{1,f}_{versicolor,1,1}, 0.4), \\ g^{f}(petalwidth, \theta^{1,f}_{versicolor,1,2}, 1.7), \\ g^{f}(petallength, \theta^{1,f}_{versicolor,1,3}, 4.9) \end{pmatrix},$$

$$Cond^{f}_{virginica,1}(x, \theta^{f}_{virginica,1}) = \min \begin{pmatrix} g^{f}(petalwidth, \theta^{1,f}_{virginica,1,1}, 0.4), \\ g^{f}(petalwidth, \theta^{1,f}_{virginica,1,2}, 1.7), \\ g^{f}(petallength, \theta^{1,f}_{virginica,1,3}, 4.9) \end{pmatrix},$$

$$Cond^{f}_{virginica,2}(x, \theta^{f}_{virginica,2}) = \min \begin{pmatrix} g^{f}(petalwidth, \theta^{1,f}_{virginica,2,1}, 0.4), \\ g^{f}(petalwidth, \theta^{1,f}_{virginica,2,2}, 1.7) \end{pmatrix}.$$

Again, the sigmoid function (fuzzy membership function) $g^{f}$is defined as increasing or decreasing.

Thus, $\theta^{f}_{setosa,1} = \{\theta^{1,f}_{setosa,1,1}, 0.4\}$,

$$\theta^{f}_{versicolor,1} = \{\theta^{1,f}_{versicolor,1,1}, 0.4, \theta^{1,f}_{versicolor,1,2}, 1.7, \theta^{1,f}_{versicolor,1,3}, 4.9\},$$

$$\theta^{f}_{virginica,1} = \{\theta^{1,f}_{virginica,1,1}, 0.4, \theta^{1,f}_{virginica,1,2}, 1.7, \theta^{1,f}_{virginica,1,3}, 4.9\}$$

and

$$\theta^{f}_{virginica,2} = \{\theta^{1,f}_{virginica,2,1}, 0.4, \theta^{1,f}_{virginica,2,2}, 1.7\}.$$

The respective fuzzy class rules are:

$$R^{f}_{setosa}(x, \theta^{f}_{setosa}) = \max(\min(g^{f}(petalwidth, \theta^{1,f}_{setosa,1,1}, 0.4))),$$

$$R^{f}_{versicolor}(x, \theta^{f}_{versicolor}) = \max\left(\min \begin{pmatrix} g^{f}(petalwidth, \theta^{1,f}_{versicolor,1,1}, 0.4), \\ g^{f}(petalwidth, \theta^{1,f}_{versicolor,1,2}, 1.7), \\ g^{f}(petallength, \theta^{1,f}_{versicolor,1,3}, 4.9) \end{pmatrix} \right)$$

and

$$R^{\mathrm{f}}_{virginica}(x,\theta^{\mathrm{f}}_{virginica}) = \max\left(\begin{array}{c} \min\left(\begin{array}{l} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,1,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,1,2},1.7),\\ g^{\mathrm{f}}(petallength,\theta^{1,\mathrm{f}}_{virginica,1,3},4.9) \end{array}\right),\\ \min\left(\begin{array}{l} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,2,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,2,2},1.7) \end{array}\right) \end{array}\right).$$

Thus,

$$\theta^{\mathrm{f}}_{setosa} = \{\theta^{\mathrm{f}}_{setosa,1}\} = \{\theta^{1,\mathrm{f}}_{setosa,1,1},0.4\},\quad \theta^{\mathrm{f}}_{versicolor} = \{\theta^{\mathrm{f}}_{versicolor,1}\}$$
$$= \{\theta^{1,\mathrm{f}}_{versicolor,1,1},0.4,\theta^{1,\mathrm{f}}_{versicolor,1,2},1.7,\theta^{1,\mathrm{f}}_{versicolor,1,3},4.9\}$$

and

$$\theta^{\mathrm{f}}_{virginica} = \{\theta^{\mathrm{f}}_{virginica,1},\theta^{\mathrm{f}}_{virginica,2}\}$$
$$= \{\theta^{1,\mathrm{f}}_{virginica,1,1},0.4,\theta^{1,\mathrm{f}}_{virginica,1,2},1.7,\theta^{1,\mathrm{f}}_{virginica,1,3},4.9,\theta^{1,\mathrm{f}}_{virginica,2,1},0.4,\theta^{1,\mathrm{f}}_{virginica,2,2},1.7\}.$$

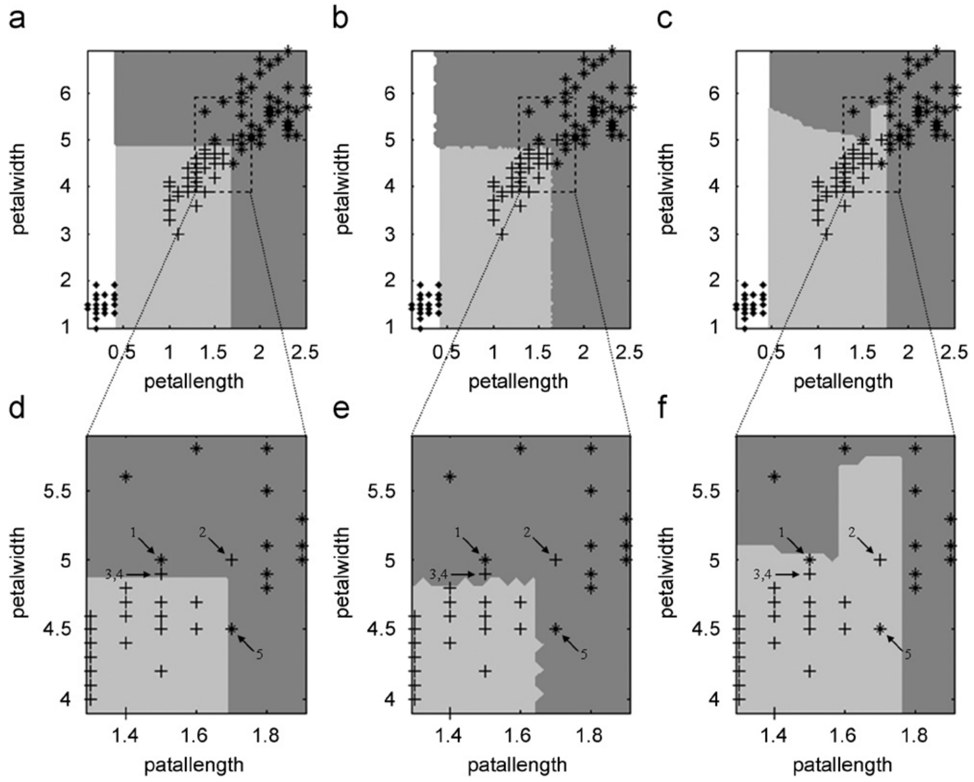

Fig. 4. Training samples plotted on the decision boundaries obtained using the training set from: (a) the decision tree, (b) the fuzzy model with the initial parameters and (c) the fuzzy model with the optimal parameters. (d), (e) and (f) are enlargements of the areas included in the dashed squares in (a), (b) and (c), respectively. White areas denote the setosa class, and the samples belonging to this class are represented with dots (◇), light gray areas denote the versicolor class, and the samples belonging to this class are represented with crosses (+). Dark gray areas denote the virginica class, and the samples belonging to this class are represented with stars (∗).

Finally, the fuzzy model M^{f} is defined as

$$M^{\mathrm{f}}(x,\Theta^{\mathrm{f}},t)=\max(t_{setosa}\cdot R^{\mathrm{f}}_{setosa},t_{versicolor}\cdot R^{\mathrm{f}}_{versicolor},t_{virginica}\cdot R^{\mathrm{f}}_{virginica})$$

$$=\max\left(\begin{array}{c} t_{setosa}\cdot\max(\min(g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{setosa,1,1},0.4))),\\ t_{versicolor}\cdot\max\left(\min\left(\begin{array}{c} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{versicolor,1,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{versicolor,1,2},1.7),\\ g^{\mathrm{f}}(petallength,\theta^{1,\mathrm{f}}_{versicolor,1,3},4.9)\end{array}\right)\right),\\ t_{virginica}\cdot\max\left(\begin{array}{c}\min\left(\begin{array}{c} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,1,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,1,2},1.7),\\ g^{\mathrm{f}}(petallength,\theta^{1,\mathrm{f}}_{virginica,1,3},4.9)\end{array}\right),\\ \min\left(\begin{array}{c} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,2,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,2,2},1.7)\end{array}\right)\end{array}\right)\end{array}\right)$$

$$=\max\left(\begin{array}{c} t_{setosa}\cdot g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{setosa,1,1},0.4),\\ t_{versicolor}\cdot\min\left(\begin{array}{c} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{versicolor,1,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{versicolor,1,2},1.7),\\ g^{\mathrm{f}}(petallength,\theta^{1,\mathrm{f}}_{versicolor,1,3},4.9)\end{array}\right),\\ t_{virginica}\cdot\min\left(\begin{array}{c} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,1,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,1,2},1.7),\\ g^{\mathrm{f}}(petallength,\theta^{1,\mathrm{f}}_{virginica,1,3},4.9)\end{array}\right),\\ t_{virginica}\cdot\min\left(\begin{array}{c} g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,2,1},0.4),\\ g^{\mathrm{f}}(petalwidth,\theta^{1,\mathrm{f}}_{virginica,2,2},1.7)\end{array}\right)\end{array}\right)$$

while,

$$\Theta^{\mathrm{f}}=\{\theta^{\mathrm{f}}_{setosa},\theta^{\mathrm{f}}_{versicolor},\theta^{\mathrm{f}}_{virginica}\}$$
$$=\left\{\begin{array}{c}\theta^{1,\mathrm{f}}_{setosa,1,1},0.4,\theta^{1,\mathrm{f}}_{versicolor,1,1},0.4,\theta^{1,\mathrm{f}}_{versicolor,1,2},1.7,\theta^{1,\mathrm{f}}_{versicolor,1,3},4.9,\\ \theta^{1,\mathrm{f}}_{virginica,1,1},0.4,\theta^{1,\mathrm{f}}_{virginica,1,2},1.7,\theta^{1,\mathrm{f}}_{virginica,1,3},4.9,\theta^{1,\mathrm{f}}_{virginica,2,1},0.4,\theta^{1,\mathrm{f}}_{virginica,2,2},1.7\end{array}\right\}$$

and $t=\{t_{setosa},t_{versicolor},t_{virginica}\}$.

A.4. Fuzzy model's parameters optimization

In order to use the Nelder–Mead method (either on local optimization strategy or as part of the hybrid optimization strategies), the initial values for the Θ^{f} and t parameters must be defined. The first parameter for each sigmoid function (which is analogous to the slope) is randomly initialized, using a normal distribution with mean value 5 and standard deviation 1: $\theta^{1,\mathrm{f}}_{i,j,k}\sim\mathrm{N}(5,1)$, while the fuzzy class rules weights $(t_{setosa},t_{versicolor},t_{virginica})$ are initialized using a uniform distribution in the [0.95, 1.05] interval: $t_i\sim\mathrm{U}(0.95,1.05)$.

In Fig. 4 we present graphically the decision boundaries obtained in each step of the realization of the methodology, for this example. Figs. 4a–c present the decision boundaries of the decision tree, the initial fuzzy model and the optimized fuzzy model, respectively, along with the training samples of each class. As it is shown, the decision boundaries of the decision tree are axis-parallel (Fig. 4a), the fuzzy model presents similar decision boundaries with the decision tree, when its parameters are initialized as described in Section 3.2 (Fig. 4b), while the decision boundaries of the optimized fuzzy model have been significantly changed. The area marked with the dashed orthogonal is enlarged in Figs. 4d–f, respectively. In Fig. 5 the same decision boundaries are shown, with the test samples of each class. The arrows in Figs. 5d–f, point to five training samples (samples 3 and 4 have the same coordinates), whose predicted class is different

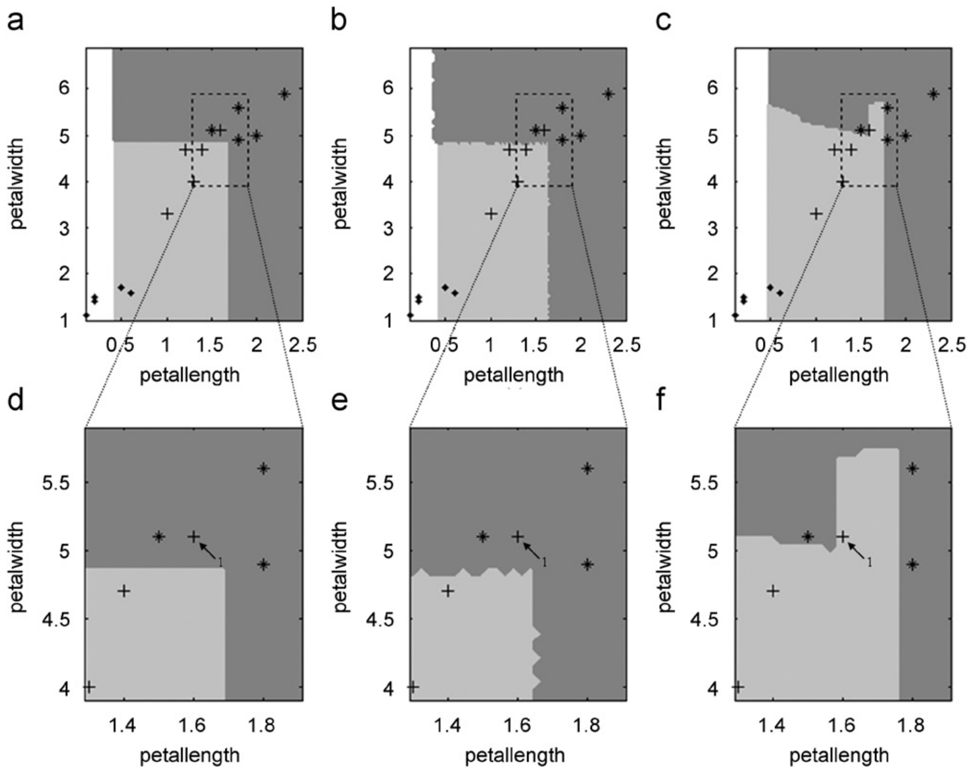

Fig. 5. Test samples plotted on the decision boundaries, obtained using the training set from: (a) the decision tree, (b) the fuzzy model with the initial parameters and (c) the fuzzy model with the optimal parameters. (d), (e) and (f) are enlargements of the areas included in the dashed squares in (a), (b) and (c), respectively. Decision areas and class samples follow the same notation as in Fig. 4.

on the optimized fuzzy model and the decision tree. More specifically, samples 2–4 (belonging to class versicolor) are misclassified by the decision tree, while samples 1 and 5 (which belong to the virginica class) are correctly classified. In the optimized fuzzy model, the decision boundaries are changed to have samples 2–4 correctly classified, while samples 1 and 5 are misclassified. Thus, the classification accuracy increases. The new decision boundaries also affect the classification accuracy of the test samples; as it is shown in Fig. 5, the test sample 1 was misclassified by the decision tree and it is correctly classified by the optimized fuzzy model, thus increasing the classification accuracy in the test set.

References

[1] J. Abonyi, H. Roubos, F. Szeifert, Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision tree initialization, Internat. J. Approx. Reasoning 4 (2003) 1–21.

[2] B. Apolloni, G. Zamponi, A.M. Zanaboni, Learning fuzzy decision trees, Neural Networks 11 (1998) 885–895.

[3] A. Asuncion, D.J. Newman, UCI machine learning repository, University of California Department of Information and Computer Science, Irvine, CA, 2007 ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩.

[4] S.M. Chen, A new approach to handling fuzzy decision making problems, IEEE Trans. Systems Man Cybernet. 18 (1988) 1012–1016.

[5] Y.-T. Chen, B. Jeng, MFILM: a multi-dimensional fuzzy inductive learning method, J. Experimental Theoret. Artificial Intelligence 17 (3) (2005) 267–281.

[6] E.K.P. Chong, S.H. Zak, An Introduction to Optimization, Wiley, New York, 2001.

[7] K. Crockett, Z. Bandar, J. O'Shea, D. Mclean, On constructing a fuzzy inference framework using crisp decision trees, Fuzzy Sets and Systems 157 (2006) 2809–2832.

[8] T.P. Exarchos, M.G. Tsipouras, C.P. Exarchos, C. Papaloukas, D.I. Fotiadis, L.K. Michalis, A methodology for the automated creation of fuzzy expert systems for ischaemic and arrhythmic beat classification based on a set of rules obtained by a decision tree, Artif. Intelligence Med. 40 (3) (2007) 187–200.

[9] J.M. Garibalti, E.I. Ifeachor, Application of simulated annealing fuzzy model tuning to umbilical cord acid-base interpretation, IEEE Trans. Fuzzy Systems 7 (1999) 72–84.
[10] Y. Goletsis, C. Papaloucas, D.I. Fotiadis, A. Likas, L.K. Michalis, Automated ischemic beat classification using genetic algorithms and multicriteria decision analysis, IEEE Trans. Biomed. Eng. 51 (10) (2004) 1717–1725.
[11] D.L. Hudson, M.E. Cohen, The role of approximate reasoning in medical expert system, in: A. Kandel (Ed.), Fuzzy Expert Systems, CRC Press, Boca Raton, FL, 1991, pp. 165–179.
[12] H. Ichihashi, T. Shirai, K. Nagasaka, T. Miyoshi, 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 and Systems 81 (1996) 157–167.
[13] C.Z. Janikow, Fuzzy decision trees: issues and methods, IEEE Trans. Systems Man Cybernet. 28 (1) (1998) 1–14.
[14] B. Jeng, Y.-M. Jeng, T.-P. Liang, FILM: a fuzzy inductive learning method for automated knowledge acquisition, Decision Support Systems 21 (1997) 61–73.
[15] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection, in: 14th Internat. Joint Conf. Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1995, pp. 1137–1143.
[16] J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, Convergence properties of the Nelder–Mead simplex method in low dimensions, SIAM J. Optim. 9 (1) (1998) 112–147.
[17] C. Olaru, L. Wehenkel, A complete fuzzy decision tree technique, Fuzzy Sets and Systems 138 (2003) 221–254.
[18] W. Pedrycz, J.V. de Oliveira, An algorithmic framework for development and optimization of fuzzy models, Fuzzy Sets and Systems 80 (1996) 37–55.
[19] C.D. Perttunen, D.R. Jones, B.E. Stuckman, Lipschitzian optimization without the Lipschitz constant, J. Optim. Theory Appl. 79 (1) (1993) 157–181.
[20] W. Pedrycz, Z.A. Sosnowski, Genetically optimized fuzzy decision trees, IEEE Trans. Systems Man Cybernet.—Part-B, Cybernet. 35 (2005) 633–641.
[21] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kauffman, California, 1993.
[22] J.R. Quinlan, Improved use of continuous attributes in C4.5, J. Artificial Intelligence Res. 4 (1996) 77–90.
[23] A. Suarez, F. Lutsko, Globally optimal fuzzy decision trees for classification and regression, IEEE Trans. Pattern Anal. Mach. Intelligence 21 (12) (1999) 1297–1311.
[24] P.N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, Reading, MA, 2005.
[25] E.C.C. Tsang, X.Z. Wang, Y.S. Yeung, Improving learning accuracy of fuzzy decision trees by hybrid neural networks, IEEE Trans. Fuzzy Systems 8 (5) (2000) 601–614.
[26] H.C. Tseng, D.W. Teo, Medical expert system with elastic fuzzy logic, 3rd IEEE Internat. Conf. on Fuzzy Systems 3 (1994) 2067–2071.
[27] M.G. Tsipouras, T.P. Exarchos, D.I. Fotiadis, Integration of global and local knowledge for fuzzy expert system creation—application to arrhythmic beat classification, in: 29th IEEE EMBS Ann. Internat. Conf. (EMBS 2007), Lyon, France, 2006, accepted for publication.
[28] M.G. Tsipouras, C. Voglis, D.I. Fotiadis, A framework for fuzzy expert system creation—application to cardiovascular diseases, IEEE Trans. Biomed. Eng. 54 (11) (2007) 2089–2105.
[29] L.H. Tsoukalas, R.E. Uhrig, Fuzzy and Neural Approaches in Engineering, Wiley, New York, 1997.
[30] L.X. Wang, A Course in Fuzzy Systems and Control, Prentice-Hall PTR, Upper Saddle River, NJ, 1997.
[31] X. Wang, B. Chen, G. Qian, F. Ye, On the optimization of fuzzy decision trees, Fuzzy Sets and Systems 112 (2000) 117–125.
[32] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, CA, USA, 2005.
[33] D.S. Yeung, E.C.C. Tsang, Weighted fuzzy production rules, Fuzzy Sets and Systems 88 (1997) 299–313.
[34] Y. Yuan, M.J. Shaw, Induction of fuzzy decision trees, Fuzzy Sets and Systems 69 (1995) 125–139.
[35] X.-J. Zeng, M.G. Singh, Approximation accuracy analysis of fuzzy systems as function approximators, IEEE Trans. Fuzzy Systems 4 (1996) 44–63.