Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Soft Computing based learning and Data Analysis: Missing Values and Data Complexity

Tesis Doctoral

Julián Luengo Martín

Granada, Diciembre de 2010

Universidad de Granada

Soft Computing based learning and Data Analysis: Missing Values and Data Complexity

MEMORIA QUE PRESENTA

Julián Luengo Martín

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Diciembre de 2010

DIRECTOR

Francisco Herrera Triguero

Departamento de Ciencias de la Computación
e Inteligencia Artificial

La memoria titulada "*Soft Computing based learning and Data Analysis: Missing and Data Complexity*", que presenta D. Julián Luengo Martín para optar al grado de doctor, ha sido realizada dentro del Máster Oficial de Doctorado "*Soft Computing y Sistemas Inteligentes*" del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección del doctor D. Francisco Herrera Triguero.

Granada, Diciembre de 2010

El Doctorando El Director

Fdo: Julián Luengo Martín Fdo: Francisco Herrera Triguero

Tesis Doctoral Parcialmente Subvencionada por el Ministerio de Educación y Ciencia bajo los Proyectos Nacionales TIN2005-08386-C05 y TIN2008-06681-C06-01. También ha sido Subvencionada bajo el Programa de Becas de Formación de Profesorado Universitario, en la Resolución del 2 de Abril de 2007, bajo la referencia AP2006-01745.

Agradecimientos

Esta memoria está dedicada a aquellas personas que han hecho posible su realización, que me han apoyado en los momentos difíciles de este largo camino y que sin ellas difícilmente hubiese llegado a buen puerto.

No puedo dejar de mencionar en primer lugar a mi familia. A mi padres Julián y Eloisa, que siempre han hecho todo lo posible para que pudiese llegar a este punto y darme todas las oportunidades que ellos siempre han deseado para mí, con abnegación y siempre una sonrisa en los buenos y malos momentos. A mi hermana Eli, que no le importa dedicarme unas risas para levantarme el ánimo, no importa lo cansado o malhumorado que esté. Un agradecimiento especial a mi pareja Laura, que siempre me ha dado ánimos aunque significase sacrificar tiempo juntos y ha comprendido la importancia que este trabajo representa para mí, siempre has sido mi apoyo. No puedo dejar de sentir que esta memoria es en parte vuestra también.

Cualquier elogio para mi director de tesis Francico Herrera es poco, no sólo por su magnífica labor en la guía y tutela de mi corto recorrido como investigador, si no por su esfuerzo y dimensión como persona. Conocerle fue un afortunado giro que no puedo dejar de agradecer, sin él esta tesis no sería lo que es, y me siento afortunado por poder contar con su apoyo en el futuro.

He tenido la suerte de conocer gente magnífica, compañeros de viaje en el que cada uno recorre su camino que a la vez compartimos todos. Me siento orgulloso de poder llamarlos amigos. En primer lugar tengo que nombrar a Salva y Alberto, que comenzaron poco antes que yo y me han dado mucho, tanto en lo académico como en lo personal. A los hermanos Jesús y Rafael Alcalá, que siempre tienen una sonrisa y buenos consejos; a Antonio Gabriel, Sergio, Javi, Daniel Molina, Alicia y Carlos Porcel siempre cercanos no importa el tiempo que haya pasado. A Óscar Cordón, Enrique Herrera, Coral del Val, José Manuel Benítez, Jorge Casillas y Manuel Lozano que con su conocimiento y experiencia siempre han estado dispuestos a facilitarme el camino. A los más jóvenes: Manolo, Nacho Pérez, Joaquín, Isaac, Nacho Robles, Victoria y Jose Antonio que empiezan su recorrido y que deseo que mantengan la ilusión y alcancen y superen sus metas. A los compañeros becarios y egresados con los que comparto magníficos recuerdos y momentos: Aída, María, Javi, Mariló, Carlos Cano, Sergio, Pedro, Fernando Bobillo, Julían Garrido, Soto y muchos más.

La vida académica te da la oportunidad de conocer muchas personas en multitud de reuniones, congresos y seminarios, y que me han ayudado asimismo a estar donde estoy. En mi caso, tengo la fortuna de poder llamar amigos a la inmensa mayoría: Cristóbal, Macarena y Rosa de Jaén; Luciano y Ana Palacios de Oviedo; Pedro Antonio y Juan Carlos de Córdoba; Ester, Albert y Nuria de Barcelona; Humberto, Edurne, Josean y Mikel de Pamplona y recientemente a Jaume Bacardit en Nottingham que me ayudó a extrañar menos mi Granada natal.

No quiero dejar de mencionar a mis amigos por lo que hemos compartido y lo que espero seguir compartiendo con ellos: Migue, Jesús, Javito, Antonio, Laurilla, José Miguel, Elena, María, Álvaro, Pepe, Diego y todos los que sabéis que disfruto con vuestra compañía y seguiré haciéndolo.

Y finalmente, una última y gran mención a todos aquellos que no aparecen explícitamente en esta memoria, pero que siempre estarán en la mía. En este trabajo podéis encontrar vuestra huella también.

GRACIAS A TODOS

Table of Contents

Part I. PhD dissertation

1. Introduction

The data acquisition and data processing is one of the hot topics in the digital world nowadays. Continuous software and hardware developments have caused that enormous amount of information is continually stored in data bases. The management and analysis of such a amount of data is beyond of human possibilities. Only the automatic processing by a computer provides the opportunity to obtain and extract useful knowledge from the stored data.

This automatic process is formerly known as Knowledge Discovery in Databases (KDD)[FHOR05, Han05, WF05] which is the non-trivial process to identify useful, new and comprehensible patterns in the data. KDD involves multiple task, in which we can include:

- Preprocessing of the data in order to correct and/or erase wrong, incomplete or inconsistent data.
- Analysis of the data in relation with the knowledge extraction task.
- Interesting patterns search using a particular representation.
- Interpretation of the extracted patterns even in a visual manner.

In this context there are many research fields which have contributed to the KDD process: data bases, pattern recognition, statistics, artificial intelligence, super-computation, etc. Artificial Intelligence has contributed to the KDD task with many techniques and resources.

The most important step in the KDD process is Data Mining (DM) [TSK05]. DM is the application of specific algorithms for extracting patterns from data and it is a multidisciplinary field. Its objective is to produce results and/or to discover relationships in the data. It can be either descriptive, i.e. to discover patterns in the data, or predictive, i.e. to predict the model's behavior based on the available data. In the first type we can find techniques such clustering, association rules or self-organizing maps. The latter type is usually referred to classification or regression algorithms. Figure 1.(a) shows the whole KDD process, while Figure 1.(b) depicts the particularization to a specific DM case.

A KDD algorithm has typically three components: the model, the preference or selection criteria and the search algorithm. The model can have two possible typologies based on its function or

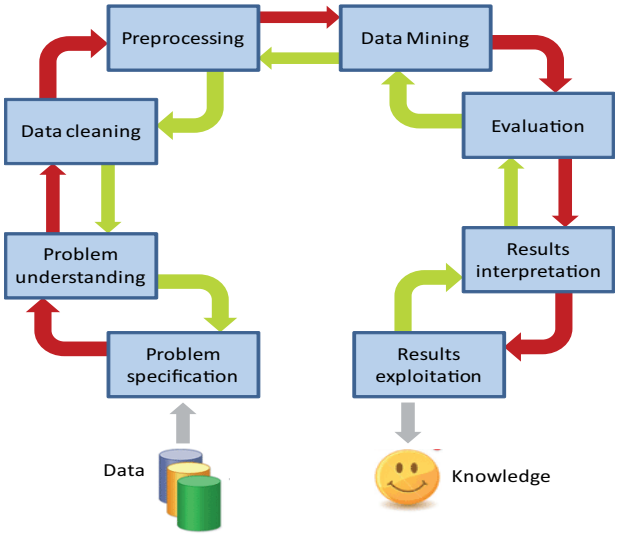

Figure 1: (a) Whole KDD process

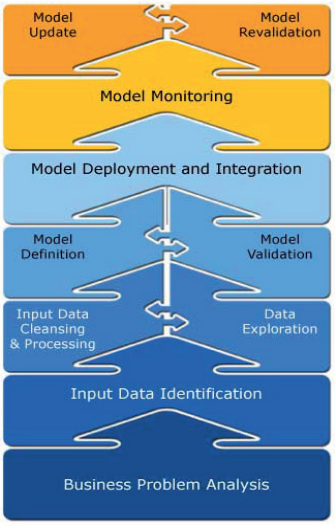

Figure 2: (b) Particular application of one Data Mining process in real world business data

representation. The first case can be classification, regression, clustering, rule generation, association rules, dependence models or sequence analysis. Based on its representation it can be artificial neural networks, decision trees, linear discrimination, support vector machines (SVMs), etc. Each model has a bunch of parameters which must be determined by a search algorithm that will optimize such models parameters based on the preference or selection criteria that better fits the model to the data.

The Machine Learning is a primary and differentiator concept with respect to the classical statistical techniques. It was conceived approximately four decades ago for developing computational methods which implement several learning forms, and in particular, mechanisms capable of induce knowledge from the data. Because software development has become a major bottleneck in computer technology today, the idea of introducing knowledge through examples seems particularly attractive. Such knowledge induction is desirable in problems with no efficient algorithmic solution, are vaguely defined, or informally specified. Examples of such problems may be medical diagnosis, marketing problems, visual pattern recognition or the detection of regularities in huge amounts of data.

Machine Learning algorithms can be divided onto two big categories:

- Black box methods like artificial neural networks or bayesian methods.
- Knowledge oriented methods, such as decision tress, association rules or decision rules.

The black box technique develops its own knowledge representation, which is not visible from the outside. The knowledge oriented models, on the contrary, build up a symbolic knowledge structure which tries to be useful from the point of view of the functionality, but also from the perspective of interpretability. There are methods to extract understandable rules from the black boxes as well, and therefore both categories can be useful for the knowledge extraction.

As we have mentioned, the DM process can be either descriptive or predictive. We enumerate the basic disciplines in each process:

- Descriptive process: clustering, association rules obtention and subgroup discovery.
- Predictive process: classification and regression.

In the context of DM we understand for classification the process in which, knowing of the existence of certain class labels or categories, we establish a rule to locate new observations in any of the existent classes (supervised learning). The classes are the product of a prediction problem, where each class corresponds to the possible output of a function. This function must be predicted from the attributes that we use to describe the elements of the data set. The necessity of a classifier arises from the requirements of having an automatic procedure, faster than a human being and able to avoid biases adopted by the expert. In this sense it allows us to avoid expensive actions in time, and to help the human experts, especially in difficult cases.

A classifier can be evaluated by five criteria:

- Accuracy: It represents the confidence level of the classifier, usually taken as the proportion of correct classification ratio that it is capable of producing.
- Speed: Classifier's response time from introducing a new example to classify, to the instant in which the classifier produces the predicted class. Usually, the speed is not as important as the accuracy.
- Interpretability: Clarity and credibility, from the human point of view, of the classification rule. The higher the interpretability of the produced model is, the more knowledge can be extracted from it.
- Learning speed: Time required by the classifier in order to obtain the classification rule from a data set.
- Robustness: Minimum number of examples needed to obtain a precise and reliable classification rule.

A classifier receives a data set as input, denoted as *training set*, and it learns the classification rule with it. In the validation process of the classifier, an extra set of examples, not used in the learning process, formerly known as *test set* is used in order to check the accuracy of the classifier.

DM techniques are very sensitive to quality of the information from which we intend to extract knowledge. The higher the quality is, the higher the quality of the obtained models will be. In this sense, the obtention of well-suited and quality data is a critical factor, and the quality of the data can be improved prior to the application of the DM process by means of the data preparation step [Pyl99]. We can consider as *preprocessing or data preparation* all those data analysis techniques which improve the data quality. Therefore the DM methods can obtain better and a bigger amount of knowledge [ZZY03]. Data preparation is relevant due to the fact that real data is usually impure, leading to the extraction of unhelpful models. Such a circumstance can be originated by missing data, noisy data or inconsistent data [KCH$^+$03]. The data preparation provides quality data, which leads to quality models. In order to do so, information retrieval, erroneous data elimination or data integration mechanisms can be used. It can also help to deal with other aspects in the data, like class imbalance, duplicate examples or reduce the data volume which leads to a better efficiency in DM.

In this memory we will focus our attention in two problems which are solved by means of data preprocessing:

- The missing data problem and how it can be solved by means of using *data imputation* preprocessing techniques. It appears when values in the data set are missing, formerly known as Missing Values (MVs).

- The class imbalance problem. It refers to those data sets in which one of the classes is described by few examples while the other is represented by many of them in comparison. It has been successfully solved by the use of *data under-sampling and over-sampling* preprocess data techniques.

A related concept to DM is Soft Computing (also known as Computational Intelligence) [Kon05]. It includes most of the methodologies that can be applied in DM. Several of the most extended and used methodologies are the genetics algorithms, fuzzy logic, neural networks, case based reasoning, rough sets or hybridizations based on all of those. Particularly, we are interested in the Fuzzy Rule Based Classification Systems (FRBCSs) [INN04] due to their high interpretability capabilities or the possibility of easily introduce and/or extract expert knowledge in their output models.

One approach to address the difficulty of the data with respect to the accuracy of a classifier is the use of data complexity measures. These measures try to capture different aspects or sources of complexity which are considered complicated to the classification task [HB06]. Therefore it is possible to establish when a problem is complicated for a classification algorithm before applying it and to act in consequence. We will pay attention to this problem along the memory, analyzing the data complexity usefulness for getting the domains of competence of different Soft Computing based learning methods.

In order to carry out this study, this memory is divided in two parts. First one is devoted to the problem statement and the discussion of the results. Second one corresponds to the publications associated to this study.

In Part I we begin by developing the problem statement introduced in this section and the techniques used to solve it with the following subsections: Subsection 1.1 introduces the problem of MVs, Subsection 1.2 describes the data complexity measures and it use, Subsection 1.3 illustrate the imbalanced classes problem and Subsection 1.4 presents the Fuzzy Rule Based Classification Systems. Next we indicate the open problems which justify the realization of this memory in Section 2 "Justification". The objectives pursued in this memory are described in Section 3 "Objectives". We include a section about the 4 "Joint Discussion of Results" which provides a summarized information about the proposals and most interesting results obtained in each part. Section 5 "Concluding Remarks: Summary of the Obtained Results and Conclusions" summarizes the results obtained in this memory and present several conclusions about them. Finally, in Section 6 "Future Works" we point out several open future works which remain open from the results of the present memory.

Finally, in order to develop the goals set, this memory is constituted by seven publications distributed in three sections which will be developed in Part II. They are the following:

- Missing Data in Classification: An Analysis on the Most Suitable Imputation Approach

- Domains of Competence of Fuzzy Rule Based Classification Systems: An Ad-hoc and an Automatic Approach

- Analysis of Over-sampling and Under-sampling approaches for Imbalanced Problems using Data Complexity Measures

1.1. Missing Values in Classification

Many existing, industrial and research data sets contain MVs. There are various reasons for their existence, such as manual data entry procedures, equipment errors and incorrect measurements. The presence of such imperfections requires a preprocessing stage in which the data is prepared and cleaned [Pyl99], in order to be useful to and sufficiently clear for the knowledge extraction process. The simplest way of dealing with missing values is to discard the examples that contain them. However, this method is practical only when the data contains a relatively small number of examples with MVs and when analysis of the complete examples will not lead to serious bias during the inference [LR87].

MVs make the performance of data analysis poor. The presence of missing values can also pose serious problems for researchers. In fact, inappropriate handling of missing data in the analysis may introduce bias and can result in misleading conclusions being drawn from a research study, and can also limit the generalizability of the research findings [WW10]. Three types of problem are usually associated with missing values in data mining [BM99]: 1) loss of efficiency; 2) complications in handling and analyzing the data; and 3) bias resulting from differences between missing and complete data.

In the case of classification, learning from incomplete data becomes even more important. Incomplete data in either the training set or test set or in both sets affect the prediction accuracy of learned classifiers [GS10]. The seriousness of this problem depends in part on the proportion of missing data. Most classification algorithms cannot work directly with incomplete data sets and due to the high dimensionality of real problems it is possible that no valid (complete) cases would be present in the data set [GLSGFV09]. Therefore, it is important to analyze which is the best technique or preprocessing considered in order to treat the present MVs before applying the classification methods as no other option is possible.

Usually the treatment of missing data in data mining can be handled in three different ways [FKP07]:

- The first approach is to discard the examples with missing data in their attributes. Therefore deleting attributes with elevated levels of missing data is included in this category too.
- Another approach is the use of maximum likelihood procedures, where the parameters of a model for the complete data are estimated, and later used for imputation by means of sampling.
- In Section 1 we saw that within DM there exist a data preprocessing step [Pyl99, ZZY03], in which the **imputation of MVs** is included. It consist of a class of procedures that aims to fill in the MVs with estimated ones. In most cases, a data set's attributes are not independent from each other. Thus, through the identification of relationships among attributes, MVs can be determined

We will focus our attention on the use of imputation methods. A fundamental advantage of this approach is that the missing data treatment is independent of the learning algorithm used. For this reason, the user can select the most appropriate method for each situation he faces. There is a wide family of imputation methods, from simple imputation techniques like mean substitution, K-Nearest Neighbour, etc.; to those which analyze the relationships between attributes such as: support vector machines-based, clustering-based, logistic regressions, maximum-likelihood procedures and multiple imputation [BM03, FKD08].

It is important to categorize the mechanisms which lead to the introduction of MVs [LR87]. The assumptions we make about the missingness mechanism and the missing data pattern of missing values can affect which imputation method could be applied, if any. As Little & Rubin [LR87] stated, there are three different mechanisms for missing data induction.

1. Missing completely at random (**MCAR**), when the distribution of an example having a missing value for an attribute does not depend on either the observed data or the missing data.
2. Missing at random (**MAR**), when the distribution of an example having a missing value for an attribute depends on the observed data, but does not depend on the missing data.
3. Not missing at random (**NMAR**), when the distribution of an example having a missing value for an attribute depends on the missing values.

In the case of the MCAR mode, the assumption is that the underlying distributions of missing and complete data are the same, while for the MAR mode they are different, and the missing data can be predicted by using the complete data [LR87]. These two mechanisms are assumed by the imputation methods so far. As stated in [FKD08] and [MPBM08], it is only in the MCAR mechanism case where the analysis of the remaining complete data (ignoring the incomplete data) could give a valid inference (classification in our case) due to the assumption of equal distributions. That is, case and attribute removal with missing data should be applied only if the missing data is MCAR, as both of the other mechanisms could potentially lead to information loss that would lead to the generation of a biased/incorrect classifier (i.e. a classifier based on a different distribution).

Another approach is to convert the missing values to a new value (encode them into a new numerical value), but such a simplistic method was shown to lead to serious inference problems [Sch97]. On the other hand, if a significant number of examples contain missing values for a relatively small number of attributes, it is beneficial to perform imputation (filling-in) of the missing values. In order to do so, the assumption of MAR randomness is needed, as Little & Rubin [LR87] observed in their analysis. In our case we will use single imputation methods, due to the time complexity of the multiple imputation schemes, and the assumptions they make regarding data distribution and MV randomness; that is, that we should know the underlying distributions of the complete data and missing data prior to their application.

1.2. Data Complexity Measures in Classification

The first intuitive indicators of the complexity of a classification problems are the number of instances in the data set, the number of features or the number of classes. However, these are very simple indicators and rarely describe all the difficulty present in the problem. More intricate issues such as the generality of the data, the inter-relationships among the variables and other factors are key for the prediction capabilities of the classifiers. An emergent field has arisen that uses a set of complexity measures [HB06] applied to quantify such particular aspects of the problem which are considered relevant to the classification task [HB02].

As a general rule, the difficulty of a problem is considered to be proportional to the error ratio obtained by a classifier. However, as the "No Free Lunch" theorem [WM97] states, it is impossible to find one best algorithm for all the problems in terms of performance. Therefore we need to establish a series of measures apart from the classification ratio in order to describe the complexity of the problem. Ho & Basu [HB02] proposed and gathered together twelve data complexity measures measures separated in three different categories:

- Measures of Overlaps in Feature Values from Different Classes: These measures are focused on the effectiveness of a single feature dimension in separating the classes, or the composite effects of a number of dimensions. They examine the range and spread of values in the data set within each class, and check for overlaps among different classes.
- Measures of Separability of Classes: These measures provide indirect characterizations of class separability. They assume that a class is made up of single or multiple manifolds that form the support of the probability distribution of the given class. The shape, position and interconnectedness of these manifolds give hints on how well two classes are separated, but they do not describe separability by design.
- Measures of Geometry, Topology and Density of Manifolds: These measures evaluate to what extent two classes are separable by examining the existence and shape of the class boundary. The contributions of individual feature dimensions are combined and summarized in a single score, usually a distance metric, rather than evaluated separately.

The data complexity measures consider geometric aspects of the data to identify the potential difficulties for classification. Figures 3, 4 and 5 depict some examples of the data complexity measures in a binary data case, indicating how different geometrical characteristics are taken into account.

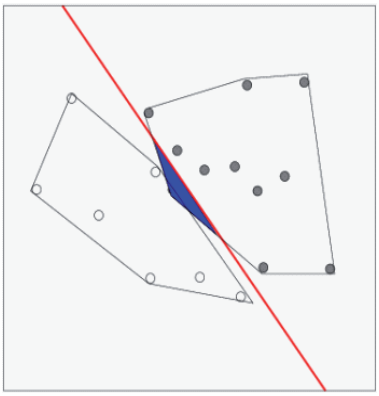

Figure 3: Degree of linear separability

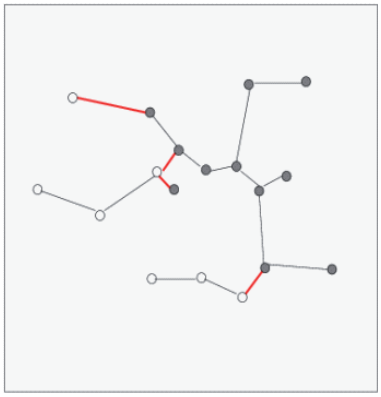

Figure 4: Length of class boundary by means of spanning trees

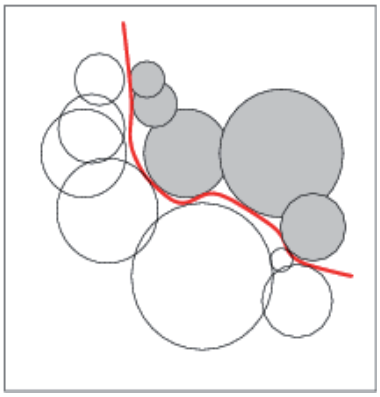

Figure 5: Shapes of class manifolds

Using these measures previous studies have characterized the strengths and weaknesses of the methods [BMH05] for a given data sets, constructing its domains of competence. The next natural step would be to indicate if a classification method is well suited for the problem and if it will perform well or bad based on these domains. This is not a new question in Machine Learning.

One of the best-known approaches to predict the classifier performance is the Meta-Learning problem, which formalized this task [BGCSV09, PBGC00, BK01a]. Meta Learning is also intended to select the best classifier for a given problem among several ones. A Meta Learning example most often involves a pair (Machine Learning problem instance, Machine Learning algorithm), labeled with the performance of the algorithm on the Machine Learning problem instance. Meta Learning faces two big problems which have hindered its prediction capabilities:

- How to represent an Machine Learning problem instance was tackled using diverse descriptors, e.g. number of examples, number of attributes, percentage of missing values, landmarkers,

etc. [PBGC00]. The difficulty is due to the fact that the descriptors must take into account the example distribution, which is not easily achieved in most cases.

- A second difficulty concerns the selection of the Machine Learning problem instances. Kalousis [Kal02] indicates that the representativity of the problems and the perturbation induce strong biases in the Meta Learning classifier.

For these reasons among others, Meta Learning has achieved limited success.

We can also refer to the least known Phase Transition approach. Using the Phase Transition, Baskiotis and Sebag [BS04] adapted the k-term DNF representation to classification problems, evaluating the C4.5 [Qui93] learning performance with respect to the underlying target concept. They defined C4.5 competence maps by means of generating boolean data sets of different characteristics (number of attributes, number of terms, etc.) based on a uniform distribution. C4.5 is then trained on these data sets and C4.5's error constitutes the complexity landscape (i.e. the competence map) using different data sets' configurations. However these competence maps are only defined for binary attributes and they are based on the assumption of a uniformly distributed sample space, which is not usually true. Furthermore, the descriptive expressions obtained are not unique, hindering their interpretability.

The data complexity measures presented can be used to carry out the characterization of the methods. One direct approach is to analyze the relationship between the data complexity value for a given data set and the performance obtained by the method. This approach does not suffer from the previous approaches' problems, as it is not dependent on a specific Machine Learning method either in the data distribution or the kind of data set attributes.

1.3. Imbalanced Classes in Classification

The problem of imbalanced classes is one of the problems that emerged when Machine Learning reached maturity, being a widely used technology in the world of business, industry and scientific research. Its importance grew as researchers realized that the analyzed data sets contained many more instances or examples from a class or classes with respect to the remaining ones [CJK04], and the obtained classification models performed below the desired threshold in the minority classes. Currently it is considered as a challenge by the Data Mining community [YW06].

Most of the learning algorithms aim to obtain a model with a high prediction accuracy and a good generalization capability. However, this inductive bias towards such a model supposes a serious challenge with the classification of imbalanced data [SWK09]. First, if the search process is guided by the standard accuracy rate, it benefits the covering of the majority examples; second, classification rules that predict the positive class are often highly specialized and thus their coverage is very low, hence they are discarded in favor of more general rules, i.e. those that predict the negative class. Furthermore, it is not easy to distinguish between noise examples and minority class examples and they can be completely ignored by the classifier.

A number of solutions have been proposed to the problem of imbalanced class in two levels: at the data level and at the algorithm level. At the data level, several re-sampling form are included, like random over-sampling with replacement, random under-sampling [EJJ04], directed over-sampling (where no new examples are created but the election of the samples to be replaced is informed rather than random), informed over-sampling with artificial examples [CBHK02], informed under-sampling [BPM04], evolutionary under-sampling [GH09] or the combination of these. At the algorithmic level, solutions include the cost-sensitive adaptations to the different classes of the problem (thus

the less represented class is the most costly one when making a prediction error), adaptation of the likelihood estimation in decision trees [WP03], decision threshold adaptation and so on.

As stated in [CCHJ08], the treatment of the imbalanced problem at the data level by means of preprocessing techniques has proven to be very useful, and it has the advantage of not needing to make any changes to the classification algorithms.

1.4. Fuzzy Rule Based Classification Systems

Fuzzy systems are one of the most important areas of application of the Fuzzy Sets Theory. In the classification framework, there is a model estructure in the form of the FRBCSs. FRBCSs constitute an extension to the classic rule-based systems, as they use rules of the form "IF-THEN". The antecedents of the rule (and in some cases, the consequents as well) consist of fuzzy logic sentences instead of classic crisp conditions. They have proven their ability for classification problems and DM in a large number of situations [Kun00, INN04].

The most common type of FRBCSs is the *linguistic* FRBCSs or *Mamdani* [Mam74] type with the following format:

R_i : IF X_{i1} IS A_{i1} AND $\cdots$ Y X_{in} IS A_{in} THEN C_k WITH PR_{ik}

where $i = 1, \ldots, M$, being X_{i1} to X_{in} the input features and C_k the output class associated to the rule, being A_{i1} to A_{in} the antecedent labels, and PR_{ik} the rule weight [IY05] (usually the certainty factor associated to the class).

In this memory we will also use FRBCSs in the form of Takagi-Sugeno Models (TSK) [TS85], which are similar to Mamdani ones, but exhibit a polynomial function as output, instead of a linguistic term.

Every FRBCSs is composed of two fundamental components: the *Knowledge Base* (KB) and the inference module. The KB is in turn composed of two elements, the *Data Base* and the *Rule Base*:

- The DB contains the linguistic terms considered in the linguistic rules and the membership functions which define the fuzzy label's semantic. In this sense, each linguistic variable included in the problem will have a fuzzy partition associated which each one of its linguistic terms. Figure 6 shows an example of a fuzzy partition with five labels.

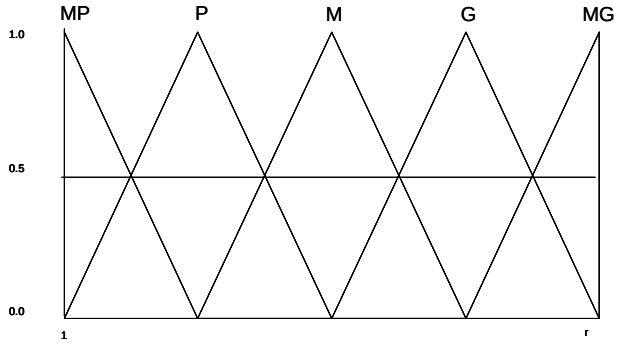

Figure 6: Fuzzy partition example

This can be considered as an approximation to the discretization for continuous domains for which we establish a membership degree to the labels, where we must include an overlapping

between them, and the inference engine handles the matching between patterns and the rules providing an output according to the consequents of the rules with positive matching. The determination of the fuzzy partitions is crucial in the fuzzy modeling [ACW06], and the granularity of the fuzzy partitions plays a fundamental role in the FRBCS' behavior [CHV00].

- The RB, composed of a collection of linguistic rules which are connected by means of a rule connective. In other words, it is possible to activate several rules simultaneously from the same input.

The module with the inference engine includes:

- A *fuzzification interface*, which transforms crisp data into fuzzy sets.
- An *inference system*, which through the data received by the fuzzification interface uses the information contained in the KB in order to perform the inference from a Fuzzy Reasoning System (FRS).

  Specifically, if we consider a new pattern $X_p = (X_{p1}, \ldots, X_{pn})$ and a RB composed of L fuzzy rules, the classification inference engine will perform the following steps [CdJH99]:

  1. *Matching degree.* To compute the *activation strength of the "IF" part for all the rules in the RB with the X_p pattern*, using a conjunction operator (usually a T-norm).

  $$\mu_{A_j}(X_p) = T(\mu_{A_{j1}}(X_{p1}), \ldots, \mu_{A_{jn}}(X_{pn})), \qquad j = 1, \ldots, L. \tag{I.1}$$

  2. *Association degree.* To compute the *association degree of pattern X_p with the M classes according to each rule in the RB.* When rules with only one consequent are considered (as presented in this section) this association degree is only referre to the class consequent of the rule ($k = C_j$).

  $$b_j^k = h(\mu_{A_j}(X_p), RW_j^k), \qquad k = 1, \ldots, M, \quad j = 1, \ldots, L. \tag{I.2}$$

  3. *Degree of consistency of the pattern classification for all classes.* We use an aggregation function which combines the positive association grades computed in the previous step.

  $$Y_k = f(b_j^k,\ j = 1, \ldots, L \text{ y } b_j^k > 0), \qquad k = 1, \ldots, M. \tag{I.3}$$

  4. *Classification.* We apply a decision function F over the consistency degree of the system for the classification pattern over all classes. This function will determine the class label l corresponding to the maximum value.

  $$F(Y_1, \ldots, Y_M) = l \quad \text{tal que} \quad Y_l = \{max(Y_k), k = 1, \ldots, M\}. \tag{I.4}$$

Finally, the generic structure of a FRBCS is shown in Figure 7.

2. Justification

Once we have presented the main concepts which this memory refers to, we pose a number of open problems which constitute the approach and justification of the current thesis memory.

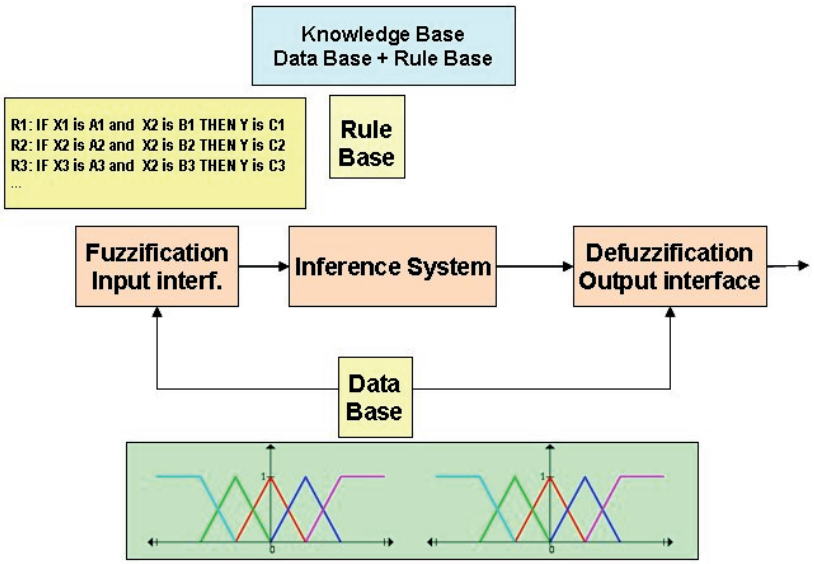

Figure 7: FRBCS structure

- As we have pointed out in Section 1.1, the presence of MVs in the data is a common issue that must be taken into account in real-world classification problems. There are many classification algorithms which have been successful, but do not consider the presence of MVs in the data. It would be too costly in time and effort to adapt them to manage MVs at the algorithmic level. In addition to this, there are different kind of classifiers based on the model they build as we have described in Section 1 and it is not clear how would be the MVs treatment be for related classifier types. This is specially important in the case of FRBCSs, which are know by being capable of handling imperfect data, but little work has been carried out to deal with MVs.

- The usual way to know if one classification method will perform well or poorly in a given problem is to run it over the data. However, large amounts of data, complex parameter configurations, slow model building and other problems may make the time necessary to obtain results too long. It is also desirable to know the reasons of why the method performs well or poorly, and if possible, in a fast and automatic process.

- As we have seen above, one of the most common techniques to deal with imbalanced class labels in classification is the use of preprocessing techniques. There are techniques based on over-sampling and under-sampling of the data which increase the accuracy of the classifier used afterwards. However, their application is not always beneficial. A first attempt to clarify this question was the Imbalance Ratio [OPBM08], defined as the ratio of the number of instances of the negative class and the positive class, but has proven to be limited to certain degree. The use of data complexity measures can help to understand this issue better.

3. Objectives

As mentioned in the previous section, the present memory is organized around three main

objectives that involve the analysis of the best imputation approach for individual and related classifiers, the diagnosis of the suitability of a classification algorithm for a classification problem prior to its application and the study of the benefits obtained by the use of over-sampling and under-sampling techniques in imbalanced data.

Specifically, the objectives that we propose are:

- *To find the best recommended imputation method for one or several classification methods.* When the missing data is found, the classification algorithms have to deal with it. Establishing the best imputation procedure in each case would save time and avoid great losses in the quality of the obtained model and its accuracy. Furthermore, knowing if the type of classifier has relevance in the imputation method chosen would ease the imputation selection process and the design of new imputation methods for specific problems or classifiers.
- *To determine the domains of competence of the classification methods based on data complexity measures.* The classification problems can have different complexity degrees depending of the classifier used afterwards. Using this complexity instead of using the classifier directly provides more information that can be used to characterize the regions in the complexity space in which the method will perform adequately or not. This fact also allows to automatically analyze classifiers jointly and to create automatic procedures to determine if a problem falls into the good or bad domain of competence of the classifier.
- *To evaluate the effect of over-sampling and under-sampling preprocessing in imbalanced classification data scenarios by means of data complexity measures.* The effects of preprocessing for imbalanced data is widely recognized as beneficial, but it has barely investigated when this occurs. The analysis of the data complexity prior to the application of the preprocessing can provide hints on the future benefits of the preprocessing. This methodology can be applied both to over-sampling and under-sampling preprocessing strategies.

4. Discussion of Results

This section shows a summary of the different proposals presented in this dissertation, and it presents a brief discussion about the obtained results by each one.

4.1. Missing Data in Classification: An Analysis on the Most Suitable Imputation Approach

In Section II.1 the problem of MVs in the classification framework and the use of imputation methods in order to overcome them is studied. The majority of Machine Learning algorithms do not consider the presence of MVs in their building process. However, the presence of MVs in the data is clearly patent and it cannot be ignored. The common approach in this context is to discard the instances which contains MVs, but this action is not drawback-free [LR87]: useful information can be discarded and data sets with high percentages of missing data can be reduced until no generalizable information is available. On the other hand, redesigning all the available algorithms is not affordable in a reasonable time. Several works have been carried out comparing new imputation

methods using classic Machine Learning algorithms like C4.5 [Qui93] or PART [FW98]. For this reason we propose to analyze the effects of imputation methods in two cases: one single method and several related classification algorithms.

In the first case we analyze the Radial Basis Function Networks. RBFNs are known to be universal approximators and to have good generalization abilities [MAC+92, Buh03]. The number of neurons selected is critical to obtain a good model for a given problem. We take into account three different RBFN approaches which cover well this aspect: a static approach [BL88] in which the number of neurons does not vary once the user specify it; a decremental [YSS97] approach, which gradually reduce the number of neurons depending from the problem; and an incremental [Pla91] approach which adds neurons when it needs to cover new regions of the problem space. One big drawback of the RBFN models is that they are not capable of managing the MVs by their own. A possible workaround is to nullify the input connecting when a MV is found [EFW01]. However, this approach is too simple and far from being adequate in the majority of problems. With this premises we analyze which is the best imputation approach for all the three variants of RBFN considered. An experimental study is included, in which the experimental framework and results are specified and a complete analysis is carried on supported by non-parametric statistical test. The obtained results indicate that from all the imputation methods analyzed, the EventCovering [WC87] stands out among all of them for the three RBFN variants with a significative margin. This means that it is possible, in some cases, to find a good imputation procedure with a good synergy with the classification method.

In the literature we can find several classification methods from the same family or which are considered related in the models they build. We have seen that it is possible to select one imputation procedure for a given classification method, but it would be also interesting to know which imputation method is the best for a given type of classification methods. If obtained, this would mean that a general recommendation can be made in advance to the use of the classifier building process. We present a classification in three big categories, *Rule Induction Learning*, *Approximate Models* and *Lazy Learning*:

- The first group is the *Rule Induction Learning* category. This group refers to algorithms which infer rules using different strategies. We consider C4.5 [Qui93], Ripper [Coh95], CN2 [CN89], AQ [MML86], PART [FW98], Slipper [CS99], SRI [PA06], Ritio [WU99] and Rule-6 [PA05] in this category.
- The second group represents the *Approximate Models*. It includes Artificial Neural Networks, Support Vector Machines and Statistical Learning. We consider C-SVM [FCL05], ν-SVM [FCL05], SMO [Pla98], RBFN [BL88], RBFND [BL88], RBFNI [Pla91], Logistic classification [lCvH92], Naïve-Bayes [DP97] and LVQ [BK01b] in this category.
- The third and last group corresponds to the *Lazy Learning* category. This group incorporates methods which do not create any model, but use the training data to perform the classification directly. We consider 1-NN and 3-NN [McL04], LWL [AMS97] and LBR [ZW00] in this category.

Many of these classifiers appear in previous studies in the literature for imputation methods, and include the previous individual study with only RBFN models.

The study carried out uses several real-wold data sets with natural MVs, indicating the followed methodology and the comparative results between classification algorithms of the same family and a global analysis based on the Wilcoxon Signed Rank Test [Dem06, GH08]. The results obtained indicate that there are always at most two or three best imputation strategies depending on the

type of classifier. The imputation quality produced is also studied based on the Wilson's Noise Ratio [Wil72] and Mutual Information measures [KC02b, KC02a, PLD05], obtaining results which are concordant with those indicated for the classifiers categories.

The FRBCSs are a special case of classifiers which are known for their interpretability and robustness to noisy data. However, in the literature are few studies of FRBCSs applied in the presence of MVs. They produce models in form or rule sets, but the nature of the rules can be linguistic (Mamdani based models) or approximative (TSK based models). Each type of FRBCSs is based on a different learning process. It could be possible that these differences in the learning process would lead to a different imputation procedures suitability. The only precedent in the literature of FRBCSs learning in the case of MVs is a technique proposed to tolerate MVs in the training of a FRBCS by Berthold & Huber [BH98]. This procedure was initially intended to estimate the best approximation to the MV based on the core region of the fuzzy label associated to the missing attribute. We present an analysis equivalent to that carried out for the classical classification methods using three different FRBCSs which cover both Mamdani and TSK types, with the same data sets and imputation methods. The results indicate that the nature of the FRBCSs plays a significant role when choosing the most suited imputation procedure, and benefits can be obtained without re-designing the inference process. It is also important to notice that the best imputation methods for FRBCSs are the same which present best values for the Wilson's Noise Ratio and Mutual Information measures.

The associated journal articles to this part are:

- J. Luengo, S. García, F. Herrera, A Study on the Use of Imputation Methods for Experimentation with Radial Basis Function Network Classifiers Handling Missing Attribute Values: The good synergy between RBFs and EventCovering method. Neural Networks 23 (2010) 406-418, doi:10.1016/j.neunet.2009.11.014
- J. Luengo, S. García, F. Herrera, On the choice of an imputation method for missing values. A study of three groups of classification methods: rule induction learning, lazy learning and approximate methods. **Submitted to Knowledge and Information Systems**.
- J. Luengo, J. Sáez, F. Herrera, Missing data imputation for Fuzzy Rule Based Classification Systems. **Submitted to Soft Computing**.

4.2. Domains of Competence of Fuzzy Rule Based Classification Systems: An Ad-hoc and an Automatic Approach

In Section II.2 the characterization of the good and bad behavior of FRBCSs by means of the use of data complexity measures is presented. The data complexity measures give information about the problem difficulty in relation with the predictive accuracy. This predicted difficulty is not related to any specific classification method, but to the problem characteristics. Therefore, if the relationship between the difficulty predicted by the data complexity measure can be established, it would be possible to predict the performance of the FRBCS in advance. Using the FH-GBML FRBCS [IYN05] which have a good average performance, we propose to obtain intervals of the data complexity metrics in which such method behaves well or poorly over a big bunch of data sets.

An initial approach is made using an ad-hoc method, evaluating the performance and the interesting intervals by the user. These intervals are constituted by the data sets in which the

FH-GBML has a prominent good or bad behavior. Therefore, it is straightforward to convert these intervals to a rule set, indicating the support of the interval, the average training and test accuracy achieved by FH-GBML. Combining these rules we are able to obtain to final rules which are capable of characterize the 75 % of the considered data sets with a differentiated behavior, constituting the domains of competence of FH-GBML.

This approach can be also applied to groups of classification algorithms with ease. We consider the use of three types of Artificial Neural Networks, Multi-Layer Perceptron [Mol93], RBFN [BL88, Buh03] and Learning Vector Quantization [BK01b]; and a Support Vector Machine [CV95, Pla98] in order to extract their domains of competence. They constitute a well-known family of related classifiers, and we intend to extract common aspects about their behavior. We apply the aforementioned methodology for all the for methods, but establishing one primordial limitation: all the extracted intervals must be present in the four methods. Therefore, we intend to extract the shared domains of competence of this family of classification algorithms. The results indicate that it is possible to do so, resulting in a characterization based on two rules which covers the 65 % of the considered data sets. Then it is possible to indicate the shared strengths and weakness of this kind of methods based on the data complexity measures.

This aforementioned approaches has some limitations:

1. The cut points which define the intervals were arbitrarily selected according to the graphics.
2. It is possible to omit intervals with similar characteristics to the extracted ones. That is, the user is not using a formal description of the good or bad behavior intervals.
3. The resultant characterization is subjective.

These issues can be tackled by the rigorous definition of the good and bad intervals, and by creating an automatic extraction method which extracts the domains of competence of the learning method. The automatic extraction method decides which data complexity measures are useful (if they contain significant intervals), and which measures are discarded (without providing any interval for them). From the obtained intervals the construction of the rules is the same as the used for the ad-hoc method. We present an extensive study based on this new automatic extraction method, overcoming these limitations using two different FRBCSs: FH-GBML (linguistic type) and PDFC (TSK type) [CW03]. We find that the characterization made by this automatic method is greatly improved in comparison with the ad-hoc approach, obtaining a characterization of 99 % of the data sets. We also validate the obtained domains of competence using a fresh bunch of data sets and compare it to related crisp classification algorithms, observing that the obtained domains of competence generalize well.

The associated journal articles to this part are:

- J. Luengo, F. Herrera, Domains of Competence of Fuzzy Rule Based Classification Systems with Data Complexity measures: A case of study using a Fuzzy Hybrid Genetic Based Machine Learning Method. Fuzzy Sets and Systems, 161 (1) (2010) 3-19 doi:10.1016/j.fss.2009.04.001.
- J. Luengo, F. Herrera, Shared Domains of Competence of Approximative Models using Measures of Separability of Classes. **Submitted to Information Sciences**.
- J. Luengo, F. Herrera, An Automatic Extraction Method of the Domains of Competence of Fuzzy Rule Based Classification Systems using Data Complexity Measures. **Submitted to IEEE Transactions on Fuzzy Systems**.

4.3. Analysis of Over-sampling and Under-sampling approaches for Imbalanced Problems using Data Complexity Measures

In Section II.3, we present the use of data complexity measures for preprocessing methods in the imbalanced data framework in order to show the effects of such preprocessing step in the data characteristics for two well-known classification algorithms: C4.5 [Qui93] and PART [FW98]. Two preprocessing approaches are considered:

- SMOTE [CBHK02] and SMOTE-ENN [BPM04] which are two over-sampling approaches.
- EUSCHC [GH09] which is one evolutionary under-sampling approach.

The Imbalance Ratio (IR) [OPBM08] is an initial guess about the difficulty of the imbalanced data, as for high IRs the use of a preprocessing technique is unavoidable. However, we can find more information if we observe the changes in the data characteristics comparing the values of the data complexity measures before and after the application of the preprocessing techniques. We present an exhaustive study over a bunch of imbalanced data sets with different IRs and data characteristics. We analyze the regions of good and bad behavior using the methodology described in Section 4.2. From the obtained results we have observed that the application of the preprocessing techniques allow to include more data sets in the region of good behavior of the method. These data sets are those which lie near to the good region border when no preprocessing is applied. Therefore we show how the application of the preprocessing methods increase the good region of C4.5 and PART. Furthermore we show that the application of the preprocessing step is not always beneficial, independently of the IR.

The associated journal articles to this part are:

- J. Luengo, A. Fernandez, S. García, F. Herrera, Addressing Data Complexity for Imbalanced Data Sets: Analysis of SMOTE-based Oversampling and Evolutionary Undersampling. Soft Computing, doi:10.1007/s00500-010-0625-8, in press (2011).

5. Concluding Remarks: Summary of the Obtained Results and Conclusions

We have addressed different problems and challenges in the MVs scope and the data complexity topic, considering the imbalanced data framework as well. In particular referring to the characterization of the performance of the classification algorithms and the analysis of the effect of imbalanced data in data complexity. We have to note that the problems tackled are independent: *missing values imputation, performance characterization by data complexity and preprocessing for imbalanced data sets evaluation by means of data complexity*; and they have been presented this way in their respective sections. We have not posed a common study in this memory and so it is raised in the future works, Section 6.

The objective is to point out that they are not connected works and for that reason we have not described a relation between the different parts. The present section briefly summarizes the obtained results and to point out the conclusions provided by this memory.

We have studied the impact of MVs in different classification algorithms, and the suitability of the use of imputation methods in order to overcome the associated problematic. We pretend to indicate the best approach or approaches in each case for a wide family of classification algorithms in order to improve the behavior of the latter based on the type of the classifier with a well founded imputation method selection. We have used a wide range of well-known classification algorithms from the literature, using the largest amount of them with respect a large selection of imputation methods and real-world data sets with natural MVs. The special case of FRBCSs is also studied, analyzed how the type of FRBCS influences in the selection of the best imputation schema, and how they are related.

The performance of individual FRBCSs is also studied based on the characteristics of the data. We have related the best regions of the data complexity with the good or bad performance of the FRBCS, creating a rule set both in an ad-hoc and an automatic way. These rule set can be summarized into two rules which describe the good and bad regions of the FRBCS. We have observed particularities in the rules obtained for each FRBCS, but also many common regions. These common regions have been exploited using a family of classification algorithms known for being closely related: artificial neural networks and SVMs. We have analyzed the common regions of good and bad behavior for these classifiers jointly, showing that they have related performance in the complexity space.

Using the same data complexity measures we have observed the effect of over-sampling and under-sampling approaches in the imbalanced data framework with respect to the C4.5 and PART algorithms. We have studied that the use of the preprocessing techniques transform the domains of competence of the classifier, widening the complexity region of good behavior and maintaining the bad behavior one. These results illustrate to what the process is beneficial and the subtle differences between the over-sampling and under-sampling approaches, being the latter more capable of producing better results.

The following sections briefly summarize the obtained results and present several conclusions.

5.1. Missing Data in Classification: An Analysis on the Most Suitable Imputation Approach

In order to solve the problems derived from the presence of MVs in the data and to improve the performance results, we have carried out an exhaustive experimental study to clarify the best imputation strategy depending on the nature of the classifier. The results are informative and significant due to the next factors:

- We have been able to find a best imputation method for a specific type of classifiers, the RBFN family, with statistical significance in both artificial and real MVs. This could be applicable to other classifiers as well.
- These recommendation has been extended to a several group of classifiers. Depending on the classifier type, we have observed different best approaches. We have corroborated this with respect to the effect of the imputation method on the data by means of two measures. Therefore we are able now to recommend an imputation procedure depending on the type of classifier used. We have checked the best imputation methods with respect to their effects in the data, and corroborated that the best imputation methods induce less noise in the preprocessed data.

- In the case of FRBCSs we have analyzed that the type of rules used (Mamdani based or TSK based) have influence in the imputation methods which should be recommended. We have observed that the TSK models have close relation with the classic approximative methods studied in the previous point.

5.2. Domains of Competence of Fuzzy Rule Based Classification Systems: An Ad-hoc and an Automatic Approach

The question of knowing if a classification algorithm is appropriate for a given problem could be solved by running the method on the problem. However, this is a trivial situation which is not always possible due to multiple questions: execution time, large parameter set configuration, many classification options available, etc. It is possible to determine if a given classification algorithm will perform well or poorly in advance by means of the use of data complexity measures or when to use a more complex parameter configuration for the difficult problems (or time-saving in the opposite case). We should also note that the data complexity measures offer information about the nature of the problem, so it is possible to improve the methods in order to improve their behavior in such difficult regions of the complexity space. The obtained results allows us to establish the domains of competence of the classifiers in different situations:

- Using twelve data complexity measures [HB02], we have characterized the domains of competence of the FH-GBML FRBCS by means of two simple rules. These rules are mutually-exclusive and separate the problems well and badly suited for FH-GBML respectively.
- The same twelve metrics have been used to extract the common domains of competence of three artificial neural networks and one SVM. These models are usually referenced in the specialized literature for being closely related. We have shown in fact that they share wide regions in the complexity space by means of only two measures of separability of classes, confirming these premises and characterizing the common problems for which these four models behave well or poorly together.
- We have proven that the characterization of the methods can be automatically done with better results. That means that the concepts of good and bad behavior can be established in the machine language and obtain even better results than when performing the same task by a human. The automation of the process has been tested on new data sets proving the generalization capabilities of the automatic extraction method, and compared to similar crisp classification methods observing particularities and shared points with them. Therefore we can characterize the classification methods in a fast and unattended manner.

5.3. Analysis of Over-sampling and Under-sampling approaches for Imbalanced Problems using Data Complexity Measures

The use of preprocessing techniques in order to tackle imbalanced problems are a common practice. However, the effects of this step has not been studied apart from their positive influence in the classification method's behavior afterwards. The IR is a hint on when the preprocessing should be applied, but we can obtain more information by means of the data complexity measures,

observing when the application of over-sampling and under-sampling techniques make the problems easier or not:

- We have obtained two final rules which are simple and precise to describe both good and bad performance of C4.5 and PART. These two rules are capable of identifying all good and bad data sets for SMOTE, SMOTE-ENN, and EUSCHC independently of the IR value.
- We show that the use of preprocessing techniques increase the region of good data sets for C4.6 and PART. Those data sets which were near to the good region but not covered when preprocessing was not applied, are covered now in the good regions if preprocessing is applied.
- An interesting consequence of the characterization obtained by the rules is that the evolutionary undersampling approach EUSCHC is capable of preprocessing successfully more data sets for C4.5 and PART.

6. Future Work

Next we present the future lines of work raised from the proposals made in this memory.

Automatic recommendation of the best imputation procedure(s)

Once we have analyzed the best approaches for the different families of classifiers, both crisp and FRBCSs, it would be possible to recommend the best imputation methods in each case. Using the typology of the classifier it is possible to assume that the best imputation methods which were the best for its type will work well. This recommendation can be wrapped in an automatic method which offers the best imputation strategy depending on the type of classifier chosen. This kind of recommendation would save huge amounts of time, as there are many imputation methods in the literature.

It can be used as well in the case of ensemble imputation methods [FKD08]. In these ensembles, several imputation methods and classifiers coexists, being the former used to impute the MVs, while the seconds are simple and fast classification algorithms used to evaluate the goodness of the imputation done. This search is no directed by any other information than the accuracy of the wrapper methods, and it can possibly enhanced using the information obtained from the studies presented in Section 4.1.

Adapting the Fuzzy Rule Based Classification Systems to use the imputed values directly

Traditionally, the presence of MVs in the data has not been considered when building up the FRBCS model. Although the FRBCS are capable of managing imperfect data, their abilities has not been explicitly checked in this case. The only precedent in the literature of FRBCSs learning in the case of MVs is a technique proposed to tolerate MVs in the training of a FRBCS [BH98]. This procedure was initially intended to estimate the best approximation to the MV based on the core region of the fuzzy label associated to the missing attribute. This initial work was further developed applying the initial technique to a particular fuzzy rule induction algorithm in [GB05]. The main idea was to avoid the use of the missing attribute in the rule operations when covering new examples or specializing the rules.

We have observed that even this primary solution cannot overcome the imputation strategies. Therefore, having observed that the Mamdani based FRBCSs benefit more from such a simple imputation strategy like Mean/Mode substitution, it is straightforward to include it at the model building. This would allow this type of FRBCSs to deal with MVs on its own with a very small increase of the model building time. In the case of TSK based models, the best imputation strategy is more complex and maybe it is not worthy to be included in the FRBCSs.

Examining the relationship of noisy and missing data for Fuzzy Rule Based Classification Systems

MVs are one particular form of data corruption and/or imprecise data, but it is not the only one [KCH+03]. Data alterations are also very common and they should have been taken into account. The quality of any data set is determined by a large number of components as described in [WSF95]. In the business world, data quality plays a critical role. Error rates in real-world data sets are common both in classes and attributes and are typically around 5 % or more [Wu96] and actions must be taken in order to avoid or mitigate the errors consequences.

FRBCSs are known to be very robust against imprecise data, but a comprehensive study about the level of the noise has not been carried out yet, while we have observed the effects of MVs at different levels. Therefore, the analysis of the behavior of FRBCSs in this framework needs to be analyzed jointly, in order to determine the capabilities of the FRBCSs in comparison with crisp classifiers.

Using the domains of competence automatic extraction method in ensemble classifiers

Ensembles of classifiers usually take into account several and simpler classifiers, which generate a final output (the chosen class for the example presented) based on several schemas. One the most widely used and known is the voting mechanism, in which each classifier choose an output class, and the most voted one is the final output. This scheme admits many variants and improvements. One of them is to assign weights to the classifiers, giving more importance to those classifiers known to be better. However, this weighting is context-free, that is, it is based on the assumption of that the classifiers with more weighting power are always the best.

We have seen using the data complexity measures that it is possible to establish the domains of competence of each classifier. Thus, using the domains of competence of every classifier considered in the ensemble, we can establish when a vote made by the classifier will be significative or not. That is, when training the ensemble classifier on a problem, using the a priori information of the data complexity measures, those base classifiers which contain such data set in their domain of competence of good behavior will have more decision power than the others.

Classifier accuracy and interpretability improving based on the data complexity measures information

The data complexity measures have a significance in their definition, as stated in [HB02]. Therefore, when a classification algorithm does not work properly in a determined region of the complexity space, we can translate it to a human understandable manner. This process was initially explored in [BMH05] indicating the strengths and weakness of the XCS classifier, but it can be taken one step further. It is possible to adequate the learning algorithm to use different learning strategies depending on the complexity values of the data set.

This procedure can be applied not only to improve the accuracy of the obtained model, but is interpretability as well in determined classification algorithms, like rule learning classifiers, FRBCSs, decision trees, etc. Indicating one measure representative of the interpretability of the model from the literature (i.e. number of rules, number of antecedents, tree depth, etc.) the relationship between the problems which produce poor interpretability and the values of data complexity can be made patent. From here, the classification algorithm can be adapted to force simpler and more interpretable models when a data set which is known to produce complex models is provided, as its complexity values can be estimated in advance.

Characterization of the preprocessing usefulness for imbalanced problems

We have observed that the data complexity measures offer information about those data sets which result more benefited from the over-sampling or under-sampling preprocessing. These regions correspond to those complexity values which where adjacent to those in which the classifier obtains good performance. On the other hand, the data sets characterized as bad for the classifier never benefit from the use of the preprocessing step.

With this information we can elaborate on the premise that some data sets will benefit from the preprocessing step, while others do not, and the complexity characteristics of each case. Therefore, with the use of the data complexity measures, we can build rules which will define the cases in which the classification algorithm will benefit from applying the preprocessing and when it will not. This would result in the characterization of the over-sampling and under-sampling preprocessing for imbalanced data sets. Such a characterization can save efforts and time, accurately indicating the cases in which preprocessing has sense.

Part II. Publications: Published, Accepted and Submitted Papers

1. Missing Data in Classification: An Analysis on the Most Suitable Imputation Approach

The journal papers associated to this part are:

1.1. A Study on the Use of Imputation Methods for Experimentation with Radial Basis Function Network Classifiers Handling Missing Attribute Values: The good synergy between RBFs and EventCovering method

- J. Luengo, S. García, F. Herrera, A Study on the Use of Imputation Methods for Experimentation with Radial Basis Function Network Classifiers Handling Missing Attribute Values: The good synergy between RBFs and EventCovering method. Neural Networks 23 (2010) 406-418, doi:10.1016/j.neunet.2009.11.014

  - Status: **Published**.
  - Impact Factor (JCR 2009): 1.879.
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 37 / 103.

Neural Networks 23 (2010) 406–418

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A study on the use of imputation methods for experimentation with Radial Basis Function Network classifiers handling missing attribute values: The good synergy between RBFNs and EventCovering method☆

Julián Luengo [a,*], Salvador García [b], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, CITIC-University of Granada, 18071, Granada, Spain

[b] Department of Computer Science, University of Jaen, 23071, Jaen, Spain

ARTICLE INFO

Article history:
Received 8 February 2009
Received in revised form 17 June 2009
Accepted 19 November 2009

Keywords:
Classification
Imputation methods
Missing values
Radial Basis Function Networks

ABSTRACT

The presence of Missing Values in a data set can affect the performance of a classifier constructed using that data set as a training sample. Several methods have been proposed to treat missing data and the one used more frequently is the imputation of the Missing Values of an instance.

In this paper, we analyze the improvement of performance on Radial Basis Function Networks by means of the use of several imputation methods in the classification task with missing values. The study has been conducted using data sets with real Missing Values, and data sets with artificial Missing Values. The results obtained show that EventCovering offers a very good synergy with Radial Basis Function Networks. It allows us to overcome the negative impact of the presence of Missing Values to a certain degree.

1. Introduction

Real data are not usually perfect – they contain wrong data, incomplete or vague (Pyle, 1999). Hence, it is usual to find missing data in most of the information sources used. There are two main reasons why an attribute value is missing: either the value was lost (e.g. it was erased) or the value was not important. The detection of incomplete data is easy in most cases: we can look for null values in the data set. However, this is not always true, since Missing Values (MV) can appear with the form of outliers or even wrong data (i.e. out of boundaries) (Pearson, 2005).

Missing data is a common problem in statistical analysis (Little & Rubin, 1987). Rates of missing data less than 1% are generally considered trivial, 1%–5% manageable. However, a rate of 5%–15% requires sophisticated methods to handle, and more than 15% may have severe impact on any kind of interpretation and harm the model's results.

Missing data treatment should be carefully thought through, otherwise bias might be introduced into the knowledge induced. Depending on the way MVs have been produced, our approach will be different. Several methods have been proposed in the literature to treat missing data (Acuna & Rodriguez, 2004; Batista & Monard, 2003). Missing data can be treated in three different ways (Li, Deogun, Spaulding, & Shuart, 2004):

- The first approach is to discard the examples with missing data in their attributes. Moreover, the case of deleting attributes with elevated levels of missing data is included in this category too.
- Other approach is the use of maximum likelihood procedures, where the parameters of a model for the complete data are estimated, and used later for impute by means of sampling.
- Finally, the imputation of MVs is a class of procedures that aims to fill in the MVs with estimated ones. In most cases, data sets attributes are not independent from each other. Thus, through the identification of relationships among attributes, MVs can be determined. This is the most commonly used approach.

Missing data has a similar impact on neural networks as it does on other types of classification algorithms, such as K-Nearest Neighbour. These similarities include variance underestimation, distribution distortion, and correlation depression. As Kros, Lin, and Brown (2006) states: "By training the network with cases containing complete data only, the internal weights developed with this type of training set cannot be accurately applied to a test set containing missing values later", and has been deeper studied in Markey, Tourassi, Margolis, and DeLong (2006). So we must impute both training and test data with the same method.

We assume that the MVs are well specified, and that we know where they appear. The study has been conducted using data sets

☆ This work was supported by the Project TIN2008-06681-C06-01. J. Luengo holds a FPU scholarship from Spanish Ministry of Education and Science.

* Corresponding author. Tel.: +34 958240598; fax: +34 958243317.
E-mail addresses: julianlm@decsai.ugr.es (J. Luengo), sglopez@ujaen.es (S. García), herrera@decsai.ugr.es (F. Herrera).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.11.014

with real MVs and data sets with artificial MVs. There is a wide family of imputation methods, from mean imputation to those which analyze the relationships between attributes. We analyze the use of different imputation strategies versus deletion case and the total lack of missing data treatment.

We will focus our attention on Radial Basis Function Networks (RBFNs), in the task of classification with MVs and the use of imputation methods for them. To this end, we present a snapshot of the state-of-the-art in the literature about MVs and Artificial Neural Networks. Moreover, a specific analysis of imputation methods for the RBFN models is the main contribution, where the EventCovering method stands out. From the results obtained we can observe that:

- Imputation methods produce significant improvements in the RBFNs results.
- A statistical analysis confirms a good synergy of RBFNs with the EventCovering method (Wong & Chiu, 1987).

The rest of the paper is organised as follows. In Section 2, we describe the imputation methods that we have used in the study and the RBFN models. Section 3 describes the experimental framework, along with the results and their analysis. Finally, in Section 4 we discuss our conclusions.

2. Preliminaries: Missing values, imputation methods and their use in Neural Networks

In this section we briefly introduce the missing data randomness, describe the imputation methods used in the study, their parameters, and present a short review on the use of imputation methods for Neural Networks. We also describe the RBFN models used in this paper.

2.1. Randomness of missing data

Depending on the reason why MVs have been produced, our approach to handle them will be different. Missing data randomness can be divided into three classes, as proposed by Little and Rubin (1987):

- Missing completely at random (*MCAR*). This is the highest level of randomness. It occurs when the probability of an instance (case) having a missing value for an attribute depends neither on the known values nor on the missing data.
- Missing at random (*MAR*). When the probability of an instance having a missing value for an attribute may depend on the known values, but not on the value of the missing data itself.
- Not missing at random (*NMAR*). When the probability of an instance having a missing value for an attribute could depend on the value of that attribute.

It is important to state the randomness of the MVs, since it will allow the use of imputation methods. In particular, we only consider *MCAR* and *MAR* situations (see Section 3.1), which are consistent with imputation techniques.

2.2. Description of imputation methods

In this subsection, we briefly describe the imputation methods that we have used. Imputation methods replace MVs with estimated values based on information available in the data set. There are many options varying from simplistic methods such as the mean imputation, to more robust methods based on relationships among attributes.

A short description of some widely used imputation methods which we have employed follows.

- Do Not Impute (*DNI*). As its name indicates, all the missing data remain un-replaced, so the networks must use their default MVs strategies. We want to verify whether imputation methods allow the Neural Networksto perform better than using the original data sets. As a guideline, we find in Grzymala-Busse and Hu (2000) a previous study of imputation methods. However, no Machine Learning method is used after the imputation process.
- Case deletion or Ignore Missing (*IM*). Using this method, all instances with at least one MV are discarded from the data set.
- Global Most Common Attribute Value for Symbolic Attributes, and Global Average Value for Numerical Attributes (*MC*) (Grzymala-Busse & Goodwin, 2005). This method is very simple: for nominal attributes, the MV is replaced with the most common attribute value; numerical values are replaced with the average of all values of the corresponding attribute.
- Concept Most Common Attribute Value for Symbolic Attributes, and Concept Average Value for Numerical Attributes (*CMC*) (Grzymala-Busse & Goodwin, 2005). As stated in *MC*, we replace the MV by the most repeated one if nominal or the mean value if numerical, but considering only the instances with same class as the reference instance.
- Imputation with K-Nearest Neighbor (*KNNI*) (Batista & Monard, 2003). Using this instance-based algorithm, every time we find a MV in a current instance, we compute the k nearest neighbors and impute a value from them. For nominal values, the most common value among all neighbors is taken, and for numerical values we will use the average value. Indeed, we need to define a proximity measure between instances. We have chosen Euclidean distance (it is a case of a L_p norm distance), which is usually used.
- Weighted imputation with K-Nearest Neighbor (*WKNNI*) (Troyanskaya et al., 2001). The Weighted K-Nearest Neighbor method selects the instances with similar values (in terms of distance) to a considered one, so it can impute as *KNNI* does. However, the estimated value now takes into account the different distances to the neighbors, using a weighted mean or the most repeated value according to the distance.
- K-means Clustering Imputation (*KMI*) (Li et al., 2004). Given a set of objects, the overall objective of clustering is to divide the data set into groups based on similarity of objects, and to minimize the intra-cluster dissimilarity. In K-means clustering, the intra-cluster dissimilarity is measured by the addition of distances among the objects and the centroid of the cluster which they are assigned to. A cluster centroid represents the mean value of the objects in the cluster. Once the clusters have converged, the last process is to fill in all the non-reference attributes for each incomplete object based on the cluster information. Data objects that belong to the same cluster are taken as nearest neighbors of each other, and we apply a nearest neighbor algorithm to replace missing data, in a way similar to that of K-Nearest Neighbor Imputation.
- Imputation with Fuzzy K-means Clustering (*FKMI*) (Acuna & Rodriguez, 2004; Li et al., 2004). In fuzzy clustering, each data object x_i has a membership function which describes the degree which this data object belongs to a certain cluster v_k. In the process of updating membership functions and centroids, we take into account only complete attributes. In this process, we cannot assign the data object to a concrete cluster represented by a cluster centroid (as done in the basic K-mean clustering algorithm), because each data object belongs to all K clusters with different membership degrees. We replace non-reference attributes for each incomplete data object x_i based on the information about membership degrees and the values of cluster centroids.

- Support Vector Machines Imputation (*SVMI*) (Feng, Chen, Yin, Yang, & Chen, 2005) is a SVM regression based algorithm to fill in missing data, i.e. set the decision attributes (output or classes) as the condition attributes (input attributes) and the condition attributes as the decision attributes, so we can use SVM regression to predict the missing condition attribute values. In order to do that, first we select the examples in which there are no missing attribute values. In the next step we set one of the condition attributes (input attribute), some of those values are missing, as the decision attribute (output attribute), and the decision attributes as the condition attributes by contraries. Finally, we use SVM regression to predict the decision attribute values.
- EventCovering (*EC*) (Wong & Chiu, 1987). Based on the work of Wong et al., a mixed-mode probability model is approximated by a discrete one. First, they discretize the continuous components using a minimum loss of information criterion. Treating a mixed-mode feature n-tuple as a discrete-valued one, the authors propose a new statistical approach for synthesis of knowledge based on cluster analysis. This method has the advantage of requiring neither scale normalization nor ordering of discrete values. By synthesis of the data into statistical knowledge, they refer to the following processes: (1) synthesize and detect from data inherent patterns which indicate statistical interdependency; (2) group the given data into inherent clusters based on these detected interdependency; and 3) interpret the underlying patterns for each clusters identified. The method of synthesis is based on author's *event–covering* approach. With the developed inference method, we are able to estimate the MVs in the data.
- Regularized Expectation-Maximization (*EM*) (Schneider, 2001). Missing values are imputed with a regularized expectation maximization (EM) algorithm. In an iteration of the EM algorithm, given estimates of the mean and of the covariance matrix are revised in three steps. First, for each record with missing values, the regression parameters of the variables with missing values on the variables with available values are computed from the estimates of the mean and of the covariance matrix. Second, the missing values in a record are filled in with their conditional expectation values given the available values and the estimates of the mean and of the covariance matrix, the conditional expectation values being the product of the available values and the estimated regression coefficients. Third, the mean and the covariance matrix are re-estimated, the mean as the sample mean of the completed data set and the covariance matrix as the sum of the sample covariance matrix of the completed data set and an estimate of the conditional covariance matrix of the imputation error. The EM algorithm starts with initial estimates of the mean and of the covariance matrix and cycles through these steps until the imputed values and the estimates of the mean and of the covariance matrix stop changing appreciably from one iteration to the next.
- Singular Value Decomposition Imputation (*SVDI*) (Troyanskaya et al., 2001). In this method, we employ singular value decomposition to obtain a set of mutually orthogonal expression patterns that can be linearly combined to approximate the values of all attributes in the data set. In order to do that, first we estimate the MVs with the *EM* algorithm, and then we compute the Singular Value Decomposition and obtain the eigenvalues. Now we can use the eigenvalues to apply a regression over the complete attributes of the instance, to obtain an estimation of the MV itself.
- Bayesian Principal Component Analysis (*BPCA*) (Oba et al., 2003). This method is an estimation method for missing values, which is based on Bayesian principal component analysis. Although the methodology that a probabilistic model and latent variables are estimated simultaneously within the framework of Bayes inference is not new in principle, the actual BPCA implementation that makes it possible to estimate arbitrary missing variables is new in terms of statistical methodology. The missing value estimation method based on BPCA consists of three elementary processes. They are (1) principal component (PC) regression, (2) Bayesian estimation, and (3) an expectation maximization (EM)-like repetitive algorithm.

Table 1
Methods parameters.

Method	Parameter
SVMI	Kernel = RBF $C = 1.0$ Epsilon = 0.001 Shrinking = No
KNNI, WKNNI	$K = 10$
KMI	$K = 10$ Iterations = 100 Error = 100
FKMI	$K = 3$ Iterations = 100 Error = 100 $m = 1.5$
EC	$T = 0.05$
EM	Iterations = 30 Stagnation tolerance = 0.0001 Inflation factor = 1 Regression type = multiple ridge regression
SVDI	Iterations = 30 Stagnation tolerance = 0.005 Inflation factor = 1 Regression type = multiple ridge regression Singular vectors = 10

A more extensive and detailed description of this method can be found in the web page http://sci2s.ugr.es/MVDM, and a PDF file with the original source paper descriptions is present in the web page formerly named "Imputation of Missing Values. Methods' Description".

There are more specialized methods, some derived from Bioinformatics. In the reviews of Farhangfar, Kurgan, and Pedrycz (2004); Grzymala-Busse and Hu (2000); Schafer and Graham (2002) we can find a good compilation of imputation methods that are not considered in this study due to their specialization.

2.3. Parameters used

In Table 1 we show the parameters used by each imputation method which has been used in this work (in the case of the method would use them). The values chosen are recommended by their respective authors.

2.4. A short review on the use of imputation methods for Neural Networks

We can find a study of the influence of MVs on Neural Networks in Ennett, Frize, and Walker (2001), where the MVs were replaced with "normal" values (i.e. replaced by zero) as well. In Yoon and Lee (1999) a specific method for training Neural Networkswith incomplete data was proposed, called Training-Estimation-Training (train with complete instances, impute the MV with the network, and train with the whole data set).

Besides, it is possible to find some work in areas related to Neural Networks. In Lim, Leong, and Kuan (2005) the authors propose a hybrid Neural Networks, in which the missing values are replaced with four Fuzzy C-Means technique based strategies.

After data set completion, the FAM module is applied. Self Organizing Maps (SOMs) are not capable of handling MV as Wang (2003) states. He proposes a SOM-based fuzzy map model for data mining with incomplete data. This model has two key components: translation of observations with missing data into fuzzy observations, and histogram-style fuzzy maps. This is not an imputation scheme, but a method which is capable of handling MVs for itself. With SVMs, Pelckmans, De Brabanterb, Suykensa, and De Moor (2005) contemplates an alternative approach where no attempt is made to reconstruct the values which are missing, but only the impact of the missingness on the outcome and the expected risk of the SVM is modeled explicitly. This is possible only when MVs are MCAR.

Artificial Neural Networks have been used as imputation methods as well in some specific applications. In Sun and Kardia (2008), the authors employ an Artificial Neural Networks based method for imputing the MVs artificially generated over genotype data. Pisoni, Pastor, and Volta (2008) also use Artificial Neural Networks for interpolating missing satellite data. A comparison between Auto-associative Neural Networks with genetic algorithm combination, and a variant of the Expectation-Maximization algorithm can be found in Nelwamondo, Mohamed, and Marwala (2007). Mileva-Boshkoska and Stankovski (2007) employ Radial Basis Function Networks for imputing ozone concentrations and SVMs as well.

2.5. Radial Basis Function Networks: Short outlook and description

The RBF architecture (Musavi, Ahmed, Chan, Faris, & Hummels, 1992) consists of a simple two layered network (a hidden layer and an output layer), each layer is fully connected to the one following.

As we have mentioned, the hidden layer is composed of a number of nodes, RBF nodes, with radial activation functions, which shall be taken, in this analysis, as Gaussian functions. Two parameters are associated with each RBF node, the "centre" and "width". Both of these quantities refer to properties of the Gaussian function. Associated with the hidden to output connections, are conventional signal multipliers: the weights. The final output processing unit merely yields a weighted sum of its inputs.

Their use in the literature is extensive, and its application varies from face recognition (Er, Wu, Lu, & Hock-Lye, 2002) to time series prediction (Harpham & Dawson, 2006). The RBFNs are under continuously research, so we can find abundant literature about extensions and improvements of RBFNs learning and modeling (Billings, Wei, & Balikhin, 2007; Ghodsi & Schuurmans, 2003; Lázaro, Santamaría, & Pantaleón, 2003; Wallace, Tsapatsoulis, & Kollias, 2005; Wei & Amari, 2008). Recently, we can find some work analyzing the behavior of RBFNs (Eickhoff & Ruckert, 2007; Liao, Fang, & Nuttle, 2003; Yeung, Ng, Wang, Tsang, & Wang, 2007) and improving their efficiency (Arenas-Garcia, Gomez-Verdejo, & Figueiras-Vidal, 2007; Schwenker, Kestler, & Palm, 2001). As we can see from the recent and past literature, we can conclude that RBFNs are a widely employed and well-known model which is actually used. Regarding MVs treatment in RBFNs there are some contributions, using the RBFNs to predict the MVs (Uysal, 2007) or obtaining the Radial Basis Function from a Vector Quantization of the data set with MVs (Lendasse, Francois, Wertz, & Verleysen, 2005). Also, the impact of MVs in the RBFNs has been considered in Morris, Boddy, and Wilkins (2001), but only in a case study.

The experimentation in this paper is conducted by using the following models of RBFNs:

- Radial Basis Function Network (*RBFN*) (Broomhead & Lowe, 1988; Buhmann, 2003). It is well suited for function approximation and pattern recognition due to its simple topological structure and its ability to reveal how learning proceeds in an explicit manner. A RBF is a function which has been built into a distance criterion with respect to a centre. Different basis functions like thin-plate spline functions, multiquadratic functions, inverse multiquadratic functions and Gaussian functions have been proposed for the hidden-layer neurons, but normally the selected one is the Gaussian function. Compared with other types of Artificial Neural Networks (ANNs), such as feed-forward networks, the RBFN requires less computation time for learning and also has a more compact topology. RBFs have been applied in the area of ANNs where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in multi-layer perceptrons. The original RBF method has been traditionally used for strict multivariate function interpolation (Powell, 1987) and for this fact, it requires as many RBF neurons as data points. Broomhead and Lowe (1988) removed this strict interpolation restriction and provided a neural network architecture where the number of RBF neurons can be far less than the data points. A RBFN mainly consists of two layers, one hidden-layer and one output layer.

  Each input example x is applied to all hidden neurons. The neuron i computes the function

$$h_i(x) = \exp\left[\frac{(x-u_i)^2}{2\sigma_i^2}\right] \tag{1}$$

  where u_i is the center of the neuron i, and h_i the output of such neuron. The RBFN has only one output (i.e. j has only one value), and it is defined by

$$z(x) = \frac{\sum_i^k h_i(x) w_i}{\sum_i^k h_i(x)}. \tag{2}$$

  The term w_i refers to the neuron weight, and k is the number of neurons in the hidden layer. In order to initialize the neurons in the network, we use a K-means clustering algorithm. The centre of the neuron is set equal to the centroid of the cluster, and its radius equal to the mean of the distance between the given center and the $N = 2$ nearest neurons:

$$\sigma_i = \sum_{j=1}^{N} \frac{d(u_i, u_j)}{N}. \tag{3}$$

  The network is initialized with a K-means clustering algorithm. In a similar fashion to the imputation method KMI, we set the number of clusters equal to the number of neurons. The initial centroids are set to random chosen examples, which are all different. By successive iterations, the neurons are adjusted using the Euclidean distance until the centroids (i.e. the neurons) do not change.

  Once the centers and radius of the neurons has been initialized, the output's weight matrix can be optimized by means of supervised training. For each train example x_i and expected output t_i, we compute the output of the hidden layer's neurons, the vector h. Next, we compute the output of the network y and compare it with the expected output t, and adjust each weight in w to reduce the Mean Square Error (MSE) with the Least Mean Squares algorithm (LMS). This method implies the use of gradient descent (delta rule) and adjusting the weights:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(t_j - y_j)h_i \tag{4}$$

  where η is the learning rate ($\eta \ll 1.0$). This process is repeated for each train example, until the max iteration limit is reached. The center and radius of the neurons is adjusted as well to minimize the output error. We use the error derivative with respect to these parameters, in a fashion similar to that of

backpropagation. With $i = 1, \ldots, m$ hidden neurons, $j = 1, \ldots, p$ inputs and $k = 1$ output, we update both parameters simultaneously from iteration n to $n + 1$ with:

$$u_{ij}(n+1) = u_ij(n) + \eta_c \frac{\left[\sum_k (t_k - y_k) w_{ik}\right] h_i (x_j - u_{ij}(n))}{(\sigma_i(n))^2} \quad (5)$$

$$\sigma_i(n+1) = \sigma_i(n) + \eta_\sigma \frac{\left[\sum_k (t_k - y_k) w_{ik}\right] h_i \parallel x - u_i(n) \parallel^2}{(\sigma_i(n))^3}. \quad (6)$$

The number of hidden-layer neurons is defined by the user a priori. In our study, we have fixed the number of neurons at 50. The η is set to 0.3. Both η_c and η_σ are set to $\frac{1}{maxIterations}$. The parameter *maxIterations* denote the maximum number of iterations of the network training algorithm, and is established to 10.

- RBFN Decremental (*RBFND*) (Yingwei, Sundararajan, & Saratchandran, 1997). In the classical approach described above, the number of hidden units is fixed a priori based on the properties of input data. A significant contribution that overcomes these drawbacks was made through the development of an algorithm that adds hidden units to the network based on the novelty of the new data. One drawback of this approach is that once a hidden unit is created, it can never be removed. The authors have proposed an algorithm that adopts the basic idea of a pruning strategy. The pruning strategy removes those hidden neurons which consistently make little contribution to the network output. Pruning becomes imperative for the identification of nonlinear systems with changing dynamics, because failing to prune the network in such cases will result in numerous inactive hidden neurons, being present as the dynamics which cause their creation initially, become nonexistent. If inactive hidden units can be detected and removed while learning proceeds, a more well-suited network topology can be constructed. Also, when the neural networks are employed for control, the problem of over-parametrization should be avoided.

  The network initialization is the same that the simple RBFN model described previously: We use a K-means algorithm to set the initial position of the neurons in the problem space. Next, we apply an initial adjust with the LMS algorithm explained in the RBFN model with 10 iterations. This procedure allow us to obtain a set of initial weights. We compute the mean $\overline{w_j}$ of the weights from all neurons for the output j as

$$\overline{w_j} = \frac{1}{N} \sum_{i=1}^{N} w_{ij}. \quad (7)$$

  Since we are using the RBFN network for classification problems, only one output is considered.

  The pruning will be applied over the neurons with a low weight compared to the computed mean. For every neuron, if its weight w_{ij} is lower than the threshold $\rho \times \overline{w_j}$, the neuron is removed from the network. The threshold ρ should be a low value. A high value will delete almost all neurons of the net, and prevent the algorithm from finish.

  Now we apply gradient descendent on the remaining neurons over u_{ij} and w_{ij}. This will re-adjust the neurons positions and their weights, trying to fill up the gaps produced by the pruning strategy.

  These combined pruning–adjusting steps are repeated over the data set and the network. When the network stays with the same neurons for λ iterations, the network training finishes.

  In our study, we provide this model with 50 initial neurons, a ρ percentage of 0.1 under the average of the weights used to decide whether a neuron must be removed or not, and a learning factor $\alpha = 0.3$ of the Least Mean Square (LMS) algorithm for adjusting the neurons. The limit iteration λ to achieve convergence has been empirically set to 15.

- RBFN Incremental (*RBFNI*) (Plat, 1991). This approach builds a RBFN composed of one hidden layer and one output layer. This topography is similar to non-incremental RBFN's one, but we do not know the number of neurons of the hidden layer. This idea is similar to RBFN Decremental, but in this model we will not set any limit to the hidden layer's neurons number. We use the Resource-Allocating Network (RAN) algorithm, which consists of a network, a strategy for allocating new units, and a learning rule for refining the network. The units on the first layer store a particular region in the input space. When the input moves away from the stored region the response of the unit decreases. As we mentioned in the RBFN model, we employ a Gaussian function to achieve this behaviour. The Eq. (1) is used to obtain the response of a single neuron, and (2) describes how to compute the network output. In Plat (1991) the author employs a default output γ parameter, which is added to the result of Eq. (1), but we do not apply it to the classification problem.

  The network starts with a blank slate: no patterns are yet stored. As patterns are presented to it, the network chooses to store some of them. At any given point the network has a current state, which reflects the patterns that have been stored previously. The allocator identifies a pattern that is not currently well represented by the network and allocates a new unit that memorizes the pattern. The output of the new unit extends to the second layer. After the new unit is allocated, the network output is equal to the desired output y. Let the index of this new unit be n.

  The peak (center) of the response of the newly allocated unit is set to the novel input example x_i:

$$u_n = x_i. \quad (8)$$

  The weight associated to this neuron to the output layer is set to the difference between the output of the network and the novel output,

$$w_n = y - z(x_i). \quad (9)$$

  The width of response (the neuron's radius) of the new unit is proportional to the distance from the nearest stored neuron to the novel input vector,

$$\sigma_n = \kappa \parallel x_j - u_{nearest} \parallel \quad (10)$$

  where κ is an overlap factor. As κ grows larger, the responses of the units overlap more and more. The RAN uses a two-part novelty condition. An input–output pair (x_j, y_j) is considered novel if the input is far away from existing centers,

$$\parallel x_j - u_{nearest} \parallel > \delta(t) \quad (11)$$

  and if the difference between the desired output and the output of the network is large

$$\parallel y_j - z(x_j) \parallel > \epsilon. \quad (12)$$

  Typically ϵ is a desired accuracy of output of the network. Errors larger than ϵ are immediately corrected by the allocation of a new unit, while errors smaller than ϵ gradually minimized using gradient descent. The distance $\delta(t)$ is the scale of resolution that the network is fitting at the tth input presentation. The learning starts with $\delta(t) = \delta_{\max}$; which is the largest length scale of interest, typically the size of the entire input space of non-zero probability density. The distance $\delta(t)$ shrinks until the it reaches $\delta_{\min}$, which is the smallest length scale of interest. The network will average over features that are smaller than $\delta_{\min}$. We used a function:

$$\delta(t) = \max(\delta_{\max} \exp(-t/\tau), \delta_{\min}) \quad (13)$$

Table 2
Data sets used for experimentation.

Data set	Abbreviation	# Ex.	# Atts.	# Classes	% MV	% Ex. with MV
Horse-colic	HOC	368	24	2	21.82	98.10
Bands	BAN	540	40	2	4.63	48.70
Hepatitis	HEP	155	20	2	5.39	48.39
House-Votes-84	HOV	434	17	2	5.30	46.54
Mammographic	MAM	961	6	2	2.81	13.63
Mushroom	MUS	8124	23	2	1.33	30.53
Autos	AUT	205	26	6	1.11	22.44
Crx	CRX	689	16	2	0.61	5.37
Post-operative	POP	90	9	3	0.37	3.33
Breast	BRE	286	10	2	0.31	3.15
Wisconsin	WIS	699	10	2	0.23	2.29
Cleveland	CLE	303	13	5	0.14	1.98
Iris+MV	IRI	150	4	3	10.00	32.67
Pima+MV	PIM	768	9	2	10.00	50.65
Wine+MV	WIN	178	14	3	10.00	70.22
Australian+MV	AUS	690	15	2	10.00	70.58
New-thyroid+MV	NTH	215	6	3	10.00	35.35
Ecoli+MV	ECO	336	8	8	10.00	48.21
Satimage+MV	SAT	6435	37	7	10.00	87.80
German+MV	GER	1000	21	2	10.00	80.00
Magic+MV	MAG	1902	11	2	10.00	58.20
Shuttle+MV	SHU	2175	10	7	10.00	55.95

where τ is a decay constant. At first, the system creates a coarse representation of the function, then refines the representation by allocating units with smaller and smaller widths. Finally, when the system has learned the entire function to the desired accuracy and length scale, it stops allocating new units altogether.

The two-part novelty condition is necessary for creating a compact network. If only condition (11) is used, then the network will allocate units instead of using gradient descent to correct small errors. If only condition (12) is used, then fine-scale units may be allocated in order to represent coarse-scale features, which is wasteful. By allocating new units the RAN eventually represents the desired function ever more closely as the network is trained. Fewer units are needed for a given accuracy if the hidden-layer outputs $h_i(x_j)$, the output-level outcome $z(x_i)$, and the thresholds γ_i are adjusted to decrease the error:

$$\Xi = \| y_j - z(x_j) \|^2 . \tag{14}$$

We use the Widrow-Hoff LMS algorithm to decrease the error whenever a new unit is not allocated, as Yingwei states:

$$\Delta z(x_j) = \alpha(y_j - z(x_j))x_j. \tag{15}$$

In addition, we adjust the centers of the responses of units to decrease error:

$$\Delta u_j = 2\frac{\alpha}{\sigma_j}(x_k - u_j)h_j(x_k)[(y_j - z(x_k)) \cdot w_j]. \tag{16}$$

Eq. (16) is derived from the gradient descent and Eq. (1). Eq. (16) also has an intuitive interpretation. Units whose outputs would cancel the error have their centers pulled towards the input. Units whose outputs would increase the error have their centers pushed away from the input.

This method tries to find the correct number of neurons for a given data set, without an initial limitation such as in the RBFN model (which has its neurons number fixed) and the RBFN Decremental model (which also has a maximum number of neurons fixed a priori). However, if δ is too low, we can find that our network overfits the training data. The model that we have used is set with $\alpha = 0.3$, $\delta_{\max} = 0.5$, $\delta_{\min} = 0.1$ and $\epsilon = 0.1$. κ is set to 1.

The parameters of the RBFN models have been empirically estimated in order to optimize the performance of RBFN in classification problems analyzed, without tuning them individually for each data set.

3. Experimental study: Imputation methods

In this section we describe the experiments we have performed. First, the data sets used and the setup of our experiments are described. Next, the obtained results are shown with an analysis of them. Finally, a specific statistical study on the behavior of EventCovering method is done. We include a graphical study of this EventCovering analysis.

3.1. Experimentation framework

We have selected a group of 22 data sets taken from the UCI repository (Asuncion & Newman, 2007). In Table 2, we summarize their properties. The column labeled as "% MV" indicates the percentage of all values of the data set which are missing. The column labeled as "% Ex. with MV" refers to the percentage of examples in the data set which have at least one MV.

Their origin is described as follows:

- We have selected 12 data sets which have MVs in a "natural" way. The percentage range of MVs varies from 20% to 0.1%. We cannot know anything about the randomness of MVs in the first 12 data sets, so we assume they are distributed in a *MAR* way.
- We have used 10 classical data sets with induced MVs (last ten rows in the Table 2). We have generated a 10% of the data set values as MVs in the training partition "artificially" in a *MCAR* way. The reason for inducing MVs only in the training partition is that we only want to discard information in the training stage, and affect the test task the least. With this configuration we can see how effective is the imputation to the unaffected instances of test.

For the experiments, we have used 10-fold cross validation. Since the training of the RBFNs is not deterministic, we have repeated each experiment 5 times per partition, each time with different seed, also randomly generated too. At the end, we have 50 executions of the model with each data set and imputation method. The imputation method uses the training partition to infer the required knowledge and relations between attributes. Then it is applied in both training and test partitions. Therefore, the test partition in each case is not used to impute the MVs.

Table 3
RBFN test accuracy

	AUT	BRE	CLE	CRX	WIS	BAN
IM	32.78 ± 10.03	**71.92 ± 7.84**	34.14 ± 10.59	66.09 ± 7.15	97.12 ± 2.50	69.41 ± 7.21
EC	41.20 ± 8.33	71.34 ± 6.45	**46.92 ± 13.01**	**84.54 ± 4.54**	**97.43 ± 2.65**	68.11 ± 4.92
KNNI	29.42 ± 8.09	70.72 ± 7.21	33.68 ± 10.68	67.18 ± 7.34	96.97 ± 2.76	71.93 ± 5.14
WKNNI	29.06 ± 7.09	69.74 ± 7.96	33.84 ± 9.63	66.11 ± 6.19	96.91 ± 2.83	71.78 ± 5.00
KMI	29.96 ± 8.87	70.76 ± 6.55	34.07 ± 9.84	65.81 ± 5.89	96.85 ± 2.86	72.26 ± 5.77
FKMI	28.46 ± 7.06	71.12 ± 7.53	33.82 ± 9.55	66.61 ± 6.43	96.97 ± 2.85	73.00 ± 4.16
SVMI	30.11 ± 6.99	71.49 ± 6.95	33.09 ± 10.06	67.30 ± 7.21	96.91 ± 2.80	**73.41 ± 5.70**
EM	**43.32 ± 8.90**	71.39 ± 5.91	34.01 ± 10.13	64.22 ± 6.01	96.82 ± 2.87	70.11 ± 4.64
SVDI	42.07 ± 9.74	71.58 ± 6.87	34.65 ± 10.22	62.10 ± 6.97	97.00 ± 2.70	70.63 ± 5.21
BPCA	40.41 ± 7.29	69.63 ± 4.65	26.32 ± 6.52	51.97 ± 6.33	45.38 ± 3.50	56.96 ± 3.79
MC	30.32 ± 7.67	70.39 ± 6.96	35.64 ± 10.54	66.90 ± 6.68	97.03 ± 2.78	72.41 ± 4.29
CMC	29.29 ± 7.37	71.63 ± 7.24	36.23 ± 10.60	66.71 ± 6.02	96.94 ± 2.83	71.89 ± 4.98
DNI	27.73 ± 7.72	71.06 ± 7.88	34.17 ± 10.51	66.81 ± 6.48	96.85 ± 2.89	71.70 ± 5.57
	HOC	HOV	MAM	POP	HEP	MUS
IM	40.00 ± 48.99	95.45 ± 4.40	79.62 ± 5.53	68.81 ± 11.22	**81.71 ± 8.99**	**99.63 ± 0.39**
EC	**79.59 ± 7.64**	95.15 ± 3.93	**82.42 ± 4.98**	70.00 ± 9.75	78.03 ± 7.57	99.59 ± 0.26
KNNI	59.50 ± 6.10	94.88 ± 3.63	79.36 ± 5.19	70.22 ± 11.43	75.24 ± 7.97	99.54 ± 0.27
WKNNI	59.55 ± 5.69	95.07 ± 3.67	78.88 ± 4.53	69.33 ± 9.57	76.03 ± 7.91	99.45 ± 0.49
KMI	59.13 ± 6.54	95.06 ± 4.04	79.15 ± 4.94	69.78 ± 10.43	77.25 ± 5.22	99.56 ± 0.30
FKMI	59.33 ± 6.10	94.97 ± 3.54	79.76 ± 4.80	68.89 ± 10.66	77.67 ± 4.52	99.57 ± 0.28
SVMI	59.06 ± 6.72	**96.12 ± 3.48**	80.98 ± 4.96	**70.44 ± 9.06**	79.89 ± 6.85	99.53 ± 0.29
EM	60.22 ± 6.90	91.86 ± 5.81	79.76 ± 5.59	68.67 ± 12.11	75.71 ± 5.55	99.44 ± 0.36
SVDI	59.84 ± 5.58	91.77 ± 5.49	79.34 ± 4.78	69.78 ± 10.90	75.13 ± 6.79	99.43 ± 0.34
BPCA	57.71 ± 6.27	49.51 ± 6.01	50.59 ± 5.39	56.44 ± 17.05	79.37 ± 2.25	50.84 ± 0.68
MC	60.17 ± 5.49	95.20 ± 3.72	79.07 ± 4.57	69.11 ± 9.76	77.35 ± 5.49	99.57 ± 0.26
CMC	59.66 ± 6.12	96.02 ± 4.17	81.13 ± 4.77	67.56 ± 11.51	77.08 ± 7.82	99.59 ± 0.21
DNI	59.07 ± 6.05	95.20 ± 3.76	79.92 ± 4.32	68.44 ± 11.19	74.78 ± 6.52	99.44 ± 0.33
	IRI	PIM	WIN	AUS	NTH	ECO
IM	90.27 ± 10.35	71.57 ± 3.69	65.29 ± 10.26	65.88 ± 5.39	87.65 ± 5.36	19.94 ± 3.87
EC	**94.40 ± 6.16**	71.89 ± 4.76	**96.52 ± 4.01**	**85.71 ± 4.39**	**92.13 ± 4.26**	**51.43 ± 13.24**
KNNI	93.87 ± 7.04	**72.49 ± 5.40**	66.69 ± 10.49	66.75 ± 5.66	90.45 ± 5.96	21.73 ± 3.11
WKNNI	92.40 ± 7.54	72.38 ± 4.83	65.65 ± 9.13	66.75 ± 4.85	88.40 ± 7.10	20.78 ± 3.67
KMI	91.73 ± 8.39	71.51 ± 5.39	67.82 ± 8.83	65.91 ± 4.77	89.48 ± 6.48	22.87 ± 2.33
FKMI	86.13 ± 10.65	71.80 ± 4.97	65.13 ± 10.77	66.29 ± 4.41	88.48 ± 6.55	22.02 ± 3.56
SVMI	94.40 ± 5.23	71.57 ± 4.67	69.79 ± 9.19	67.13 ± 4.85	90.82 ± 5.36	22.40 ± 3.38
EM	89.73 ± 8.45	71.24 ± 5.29	65.41 ± 9.33	63.36 ± 4.45	89.68 ± 6.38	25.43 ± 8.58
SVDI	88.53 ± 10.59	71.68 ± 4.43	62.88 ± 8.86	61.19 ± 4.42	89.02 ± 6.45	23.49 ± 5.62
BPCA	33.47 ± 1.63	65.04 ± 4.67	28.10 ± 2.57	51.10 ± 5.54	65.15 ± 3.18	22.15 ± 2.58
MC	94.27 ± 5.81	71.94 ± 5.44	68.12 ± 5.94	66.41 ± 5.29	88.69 ± 5.93	22.23 ± 2.71
CMC	94.40 ± 5.87	71.62 ± 4.68	67.25 ± 7.98	66.55 ± 4.67	90.93 ± 5.74	21.91 ± 2.89
DNI	91.47 ± 7.66	71.85 ± 4.53	67.55 ± 10.20	66.29 ± 5.72	89.52 ± 6.10	20.68 ± 3.80
	SAT	GER	MAG	SHU		
IM	59.97 ± 4.77	62.54 ± 5.62	75.68 ± 3.37	88.81 ± 2.83		
EC	62.67 ± 4.69	**72.26 ± 3.39**	**77.13 ± 3.53**	**97.26 ± 3.00**		
KNNI	67.34 ± 2.73	67.52 ± 4.31	76.35 ± 2.85	90.26 ± 3.83		
WKNNI	67.00 ± 3.01	68.26 ± 3.75	76.10 ± 2.55	90.14 ± 3.62		
KMI	68.29 ± 2.95	67.96 ± 3.55	75.86 ± 3.37	89.77 ± 2.03		
FKMI	**69.68 ± 2.16**	68.24 ± 3.42	75.84 ± 3.02	89.30 ± 4.45		
SVMI	69.38 ± 2.74	67.44 ± 4.03	76.37 ± 2.74	89.45 ± 4.31		
EM	17.28 ± 8.76	68.52 ± 3.45	74.11 ± 3.31	82.54 ± 5.13		
SVDI	19.02 ± 8.15	68.12 ± 3.53	73.97 ± 3.01	82.06 ± 3.42		
BPCA	16.53 ± 0.79	64.12 ± 5.19	61.96 ± 1.54	62.90 ± 3.61		
MC	69.41 ± 2.83	67.92 ± 3.89	75.76 ± 2.58	87.81 ± 3.70		
CMC	69.46 ± 2.30	66.60 ± 3.54	76.39 ± 2.97	88.95 ± 4.33		
DNI	45.54 ± 5.22	67.36 ± 4.03	76.51 ± 2.97	86.50 ± 4.47		

3.2. Experiments and analysis

In this section we present the experimental results, and analyze them. We have summarized the percentage of well-classified instances in test. We have 50 runs of each problem (5 times per partition), the mean of these 50 experiments is shown as a representative value, and the standard deviations have been computed. We present the obtained results in Tables 3–5, a table per each RBFN model. The highest accuracy value is represented in bold, emphasizing the best method of a given data set.

From our study, we can point out the following:

- *DNI* (Do Not Impute) method is almost never the best method. This method informs us about the relevance of MV in the attributes.
- *IM* (Ignore instances with MVs) method has a very poor performance as well. Sometimes, the deletion of instances leads the RBFN method not to adjust itself to certain classes, since the information that describes them has been erased from the data set. In these situations, the results of the test accuracy are even poorer than the *DNI* method. However, data sets with a low percentage of MVs show a low disadvantage of this method from the rest.
- Simple methods as *MC* and *CMC* are competitive and can surpass more sophisticated methods. However, the use of these methods introduces bias in the data, thus the RBFN model can be penalized.
- The clustering methods based on K-Means and K-NN have an average performance, not always better than *DNI* or

Table 4
RBFN decremental test accuracy.

	AUT	BRE	CLE	CRX	WIS	BAN
IM	29.09 ± 12.13	62.01 ± 10.92	34.99 ± 13.70	59.32 ± 8.15	88.15 ± 11.11	56.03 ± 10.32
EC	**46.82 ± 13.71**	62.16 ± 10.42	**43.78 ± 15.53**	**76.39 ± 7.57**	**92.73 ± 6.03**	58.74 ± 8.94
KNNI	25.41 ± 11.22	62.89 ± 11.00	34.00 ± 11.47	59.72 ± 6.40	86.41 ± 12.50	60.41 ± 9.42
WKNNI	25.02 ± 9.51	64.50 ± 8.40	31.52 ± 11.34	59.16 ± 7.59	85.00 ± 16.21	60.70 ± 9.03
KMI	25.08 ± 10.49	62.53 ± 7.85	34.87 ± 12.51	59.29 ± 8.02	87.58 ± 11.75	62.81 ± 5.92
FKMI	25.18 ± 9.91	62.92 ± 9.49	33.61 ± 11.61	59.56 ± 7.86	88.27 ± 10.18	60.52 ± 8.45
SVMI	25.07 ± 9.77	64.54 ± 9.06	35.02 ± 12.41	60.79 ± 7.06	91.71 ± 5.46	62.67 ± 6.38
EM	33.87 ± 12.08	64.76 ± 8.90	34.90 ± 12.63	58.18 ± 7.29	91.13 ± 5.45	60.67 ± 7.06
SVDI	33.73 ± 12.35	62.95 ± 9.64	33.98 ± 11.91	59.81 ± 7.68	86.84 ± 10.99	63.00 ± 8.66
BPCA	29.87 ± 11.00	**70.99 ± 8.52**	33.08 ± 9.61	49.61 ± 6.23	46.33 ± 5.69	50.67 ± 6.97
MC	26.91 ± 11.38	62.84 ± 10.30	35.05 ± 11.31	57.11 ± 8.86	87.23 ± 10.15	60.44 ± 9.01
CMC	29.19 ± 9.42	65.19 ± 8.01	35.83 ± 11.44	59.46 ± 8.65	88.42 ± 11.98	**63.00 ± 8.41**
DNI	23.44 ± 11.43	62.86 ± 10.61	38.40 ± 12.30	60.42 ± 7.80	90.79 ± 8.53	59.30 ± 8.51
	HOC	HOV	MAM	POP	HEP	MUS
IM	40.00 ± 48.99	87.98 ± 10.74	78.15 ± 5.49	59.89 ± 16.53	**81.20 ± 10.05**	84.26 ± 15.62
EC	**66.80 ± 11.15**	88.07 ± 11.54	78.94 ± 5.29	58.00 ± 15.76	74.45 ± 11.95	86.59 ± 15.00
KNNI	60.98 ± 5.66	88.92 ± 9.33	76.49 ± 4.75	52.00 ± 17.84	62.01 ± 16.16	86.71 ± 13.28
WKNNI	59.46 ± 5.30	87.40 ± 13.08	76.34 ± 6.25	59.33 ± 16.72	65.09 ± 15.24	85.93 ± 13.89
KMI	58.58 ± 6.24	86.69 ± 10.86	76.49 ± 5.84	**60.89 ± 13.74**	63.74 ± 15.60	86.31 ± 10.61
FKMI	60.87 ± 6.01	**90.21 ± 9.65**	77.44 ± 5.22	56.67 ± 16.37	66.19 ± 12.51	84.14 ± 14.29
SVMI	58.55 ± 6.69	88.14 ± 12.66	77.55 ± 5.34	53.78 ± 17.69	71.37 ± 15.76	**88.40 ± 11.39**
EM	61.20 ± 4.73	83.37 ± 12.07	77.38 ± 5.46	58.44 ± 14.37	60.93 ± 18.23	85.56 ± 11.71
SVDI	60.75 ± 7.12	84.43 ± 12.08	77.61 ± 5.94	56.00 ± 14.22	62.68 ± 15.89	86.63 ± 12.34
BPCA	55.11 ± 7.50	50.99 ± 6.38	51.03 ± 4.71	46.67 ± 17.36	70.33 ± 12.54	50.81 ± 1.17
MC	60.72 ± 6.22	83.09 ± 16.46	76.55 ± 5.00	54.44 ± 18.36	64.97 ± 13.58	86.64 ± 12.25
CMC	59.41 ± 6.23	85.75 ± 14.19	**79.25 ± 4.74**	57.11 ± 16.48	69.64 ± 17.33	86.85 ± 13.80
DNI	58.86 ± 6.04	86.33 ± 11.81	76.82 ± 4.89	60.44 ± 15.74	66.67 ± 13.93	86.39 ± 13.15
	IRI	PIM	WIN	AUS	NTH	ECO
IM	92.93 ± 6.02	63.44 ± 9.37	66.84 ± 11.65	57.48 ± 6.02	77.30 ± 12.25	38.94 ± 11.65
EC	**94.40 ± 5.05**	**68.74 ± 5.31**	**89.75 ± 9.91**	**78.35 ± 10.44**	82.77 ± 16.19	43.38 ± 13.38
KNNI	93.73 ± 5.40	67.92 ± 6.36	66.89 ± 12.66	57.83 ± 7.57	80.01 ± 12.33	41.53 ± 7.54
WKNNI	92.27 ± 6.02	65.79 ± 6.77	66.01 ± 11.90	58.29 ± 7.28	79.52 ± 12.01	38.37 ± 11.02
KMI	91.47 ± 7.55	67.51 ± 7.36	68.90 ± 11.56	57.80 ± 6.95	79.60 ± 11.16	37.85 ± 12.02
FKMI	92.93 ± 6.72	66.04 ± 6.93	64.65 ± 12.31	56.06 ± 6.76	78.95 ± 10.81	39.04 ± 11.13
SVMI	94.27 ± 5.50	63.98 ± 8.21	69.20 ± 10.80	58.78 ± 6.09	81.76 ± 10.67	41.06 ± 10.42
EM	90.27 ± 9.35	66.19 ± 9.48	52.78 ± 13.09	54.46 ± 8.13	78.12 ± 15.60	40.02 ± 9.17
SVDI	91.07 ± 7.61	67.11 ± 5.93	57.88 ± 12.85	55.97 ± 6.86	77.61 ± 13.88	**44.46 ± 9.24**
BPCA	33.33 ± 0.00	66.59 ± 8.31	28.78 ± 3.51	49.65 ± 5.55	64.86 ± 8.15	38.44 ± 7.69
MC	94.27 ± 6.11	65.24 ± 5.55	64.04 ± 12.96	58.64 ± 7.72	**83.31 ± 13.40**	39.07 ± 11.14
CMC	93.73 ± 5.56	66.12 ± 7.74	67.85 ± 10.30	59.25 ± 7.22	81.03 ± 9.78	42.29 ± 8.22
DNI	92.40 ± 8.11	63.93 ± 9.11	63.63 ± 10.17	56.49 ± 7.41	78.46 ± 12.87	36.42 ± 14.05
	SAT	GER	MAG	SHU		
IM	33.25 ± 8.05	60.26 ± 5.12	64.56 ± 9.94	75.42 ± 17.14		
EC	35.55 ± 12.40	63.32 ± 7.26	**69.09 ± 10.17**	**87.45 ± 11.38**		
KNNI	38.47 ± 9.37	**68.68 ± 2.49**	63.74 ± 10.47	76.42 ± 16.53		
WKNNI	36.84 ± 9.26	68.04 ± 3.19	62.75 ± 10.38	74.02 ± 16.85		
KMI	37.93 ± 10.11	67.54 ± 4.15	63.95 ± 9.63	69.13 ± 18.36		
FKMI	33.50 ± 9.02	66.92 ± 5.44	64.90 ± 8.77	73.45 ± 15.11		
SVMI	**38.82 ± 10.82**	67.10 ± 3.67	64.46 ± 8.64	73.73 ± 12.91		
EM	23.63 ± 10.36	67.10 ± 3.40	64.13 ± 8.11	66.84 ± 16.96		
SVDI	24.36 ± 10.27	66.50 ± 6.01	64.59 ± 9.26	61.22 ± 21.72		
BPCA	15.63 ± 3.04	64.40 ± 8.63	57.78 ± 7.18	50.98 ± 12.03		
MC	36.86 ± 10.08	66.80 ± 5.39	62.58 ± 9.75	70.51 ± 14.42		
CMC	37.35 ± 8.98	67.86 ± 3.14	64.74 ± 8.36	78.16 ± 12.78		
DNI	30.43 ± 9.83	67.98 ± 6.73	62.66 ± 11.28	65.05 ± 19.37		

IM, and sometimes surpassed by simple methods like mean substitution.

- *SVMI* method does not perform very well, despite of the RBF Kernel we have chosen. This method appears to be very dependent on the data set.
- *EM* method does not perform as well as the best method, but it also never leads to the worse accuracy. Compared to *SVDI*, it obtains better results, although *SVDI* is based on *EM* imputation method.
- *BPCA* has a very poor performance sometimes. Except for RBFN Incremental in certain cases, their results are below the average.
- Finally, *EC* is the best method in many data sets. It is the best 13 times for RBFN, 11 times for RBFND and 13 times for RBFNI. It offers good stability in each data set, and can surpass the accuracy in more than 10% in some data sets respect to other imputation methods.

As a summary from the previous analysis we can observe that *DNI* is not the best option, since it can be improved by many methods, which are capable of capture the relationships between values of the instance. The use of case deletion (i.e. *IM*) is discouraged too, since it rarely outperforms the results of the imputation methods, and can seriously harm the test accuracy.

Table 5
RBFN incremental test accuracy.

	AUT	BRE	CLE	CRX	WIS	BAN
IM	36.69 ± 10.01	66.45 ± 8.48	35.33 ± 10.74	62.94 ± 5.38	96.02 ± 3.31	74.02 ± 7.61
EC	67.65 ± 9.77	61.75 ± 7.87	**54.00 ± 8.05**	**81.49 ± 4.08**	**96.23 ± 2.71**	**75.96 ± 5.97**
KNNI	34.52 ± 8.59	63.76 ± 7.13	35.89 ± 8.07	63.38 ± 5.18	95.66 ± 2.75	74.93 ± 6.43
WKNNI	34.56 ± 12.16	65.83 ± 8.87	35.46 ± 9.40	62.28 ± 5.95	95.77 ± 2.80	75.59 ± 5.96
KMI	34.23 ± 12.90	65.05 ± 6.19	34.61 ± 9.00	63.90 ± 4.93	96.20 ± 2.52	75.59 ± 6.30
FKMI	35.27 ± 11.66	65.63 ± 7.37	34.59 ± 9.96	63.69 ± 5.17	95.54 ± 4.02	74.70 ± 5.18
SVMI	36.60 ± 11.01	64.72 ± 7.88	35.74 ± 7.78	63.56 ± 6.29	96.20 ± 2.61	75.22 ± 6.05
EM	46.58 ± 10.17	64.33 ± 7.10	35.57 ± 9.61	63.15 ± 5.72	96.03 ± 2.46	75.41 ± 5.34
SVDI	46.56 ± 12.12	64.16 ± 8.27	35.75 ± 9.78	61.23 ± 5.42	95.88 ± 3.13	75.74 ± 6.57
BPCA	**76.96 ± 9.51**	**94.39 ± 3.85**	37.48 ± 8.95	51.97 ± 3.42	45.21 ± 3.25	57.41 ± 5.98
MC	34.45 ± 10.05	63.42 ± 8.13	36.10 ± 8.89	63.75 ± 5.44	95.83 ± 2.92	75.15 ± 5.12
CMC	36.72 ± 10.80	63.62 ± 7.37	36.60 ± 8.22	64.42 ± 5.74	95.85 ± 2.46	75.67 ± 6.73
DNI	32.52 ± 10.81	65.86 ± 7.36	35.36 ± 9.46	63.41 ± 5.64	96.00 ± 3.04	75.78 ± 4.88
	HOC	HOV	MAM	POP	HEP	MUS
IM	40.00 ± 48.99	95.42 ± 3.90	76.19 ± 4.49	59.44 ± 17.51	**77.70 ± 13.04**	**100.00 ± 0.00**
EC	**75.71 ± 6.13**	94.78 ± 4.04	**79.46 ± 6.15**	60.00 ± 15.56	75.33 ± 11.08	**100.00 ± 0.02**
KNNI	57.41 ± 5.70	94.23 ± 4.06	75.40 ± 4.20	59.11 ± 16.40	58.65 ± 13.77	**100.00 ± 0.00**
WKNNI	56.40 ± 8.25	94.50 ± 4.37	75.65 ± 4.41	**62.67 ± 16.14**	61.15 ± 14.23	**100.00 ± 0.02**
KMI	60.84 ± 8.31	94.64 ± 3.73	75.19 ± 4.85	60.89 ± 13.74	61.41 ± 13.15	99.99 ± 0.03
FKMI	58.02 ± 8.21	94.46 ± 3.65	75.73 ± 4.83	60.00 ± 16.33	58.11 ± 13.04	**100.00 ± 0.02**
SVMI	55.94 ± 7.59	94.73 ± 4.07	76.41 ± 4.57	60.89 ± 16.52	74.87 ± 13.53	**100.00 ± 0.00**
EM	60.12 ± 7.37	90.76 ± 6.33	75.40 ± 5.09	60.00 ± 17.07	63.25 ± 13.89	**100.00 ± 0.00**
SVDI	57.90 ± 6.99	90.66 ± 5.89	76.20 ± 4.53	58.89 ± 15.60	67.30 ± 12.19	99.99 ± 0.04
BPCA	54.59 ± 5.09	50.20 ± 7.36	50.11 ± 4.39	43.78 ± 17.41	73.67 ± 3.25	50.84 ± 0.68
MC	57.90 ± 6.82	94.64 ± 4.16	74.97 ± 4.09	60.22 ± 17.08	62.88 ± 13.56	**100.00 ± 0.02**
CMC	60.10 ± 8.51	**95.66 ± 3.76**	76.92 ± 5.13	58.44 ± 16.60	73.21 ± 10.39	99.99 ± 0.03
DNI	57.96 ± 8.16	94.18 ± 3.67	76.05 ± 4.36	62.00 ± 16.04	62.19 ± 13.19	**100.00 ± 0.00**
	IRI	PIM	WIN	AUS	NTH	ECO
IM	94.67 ± 5.66	65.98 ± 6.38	64.87 ± 11.53	59.68 ± 5.55	87.10 ± 7.87	50.34 ± 12.45
EC	94.13 ± 4.92	**70.26 ± 4.73**	**87.06 ± 17.67**	**82.43 ± 4.79**	90.91 ± 5.09	**67.70 ± 7.18**
KNNI	94.67 ± 4.42	67.44 ± 6.70	71.40 ± 8.20	59.83 ± 6.72	87.80 ± 8.28	54.26 ± 10.69
WKNNI	94.53 ± 4.36	65.08 ± 7.59	70.65 ± 8.89	60.72 ± 5.89	87.67 ± 6.69	54.40 ± 10.10
KMI	95.20 ± 5.17	65.94 ± 6.37	68.90 ± 9.18	58.81 ± 6.77	87.36 ± 7.65	54.16 ± 12.20
FKMI	95.20 ± 5.34	65.83 ± 5.57	69.92 ± 10.18	59.57 ± 6.14	88.02 ± 7.03	49.53 ± 11.89
SVMI	94.67 ± 6.11	64.72 ± 5.87	72.65 ± 8.85	59.25 ± 5.91	88.84 ± 6.05	57.06 ± 10.95
EM	94.67 ± 5.96	67.31 ± 5.13	56.69 ± 13.34	59.97 ± 5.38	86.32 ± 7.70	52.04 ± 10.02
SVDI	94.00 ± 7.45	64.56 ± 6.03	58.42 ± 14.04	60.00 ± 5.55	86.08 ± 8.08	51.65 ± 10.10
BPCA	33.33 ± 0.00	99.19 ± 0.82	33.06 ± 2.10	47.80 ± 6.56	69.62 ± 1.86	35.48 ± 5.83
MC	95.60 ± 4.54	63.93 ± 6.10	72.66 ± 9.29	59.74 ± 5.73	86.37 ± 7.72	50.87 ± 12.55
CMC	94.67 ± 5.33	67.30 ± 5.23	72.83 ± 9.31	61.10 ± 5.78	**90.93 ± 5.89**	56.11 ± 9.75
DNI	**95.73 ± 4.57**	66.73 ± 6.69	68.51 ± 10.59	60.17 ± 5.24	88.19 ± 6.47	53.55 ± 13.18
	SAT	GER	MAG	SHU		
IM	60.94 ± 3.16	53.98 ± 5.00	71.42 ± 3.63	93.23 ± 3.47		
EC	76.82 ± 1.51	**66.74 ± 5.18**	**76.20 ± 3.97**	**97.57 ± 5.32**		
KNNI	76.46 ± 2.74	55.64 ± 5.42	71.03 ± 3.25	95.42 ± 2.50		
WKNNI	76.86 ± 2.50	57.30 ± 4.79	72.84 ± 3.20	95.58 ± 1.54		
KMI	77.03 ± 2.82	56.06 ± 6.03	71.22 ± 3.39	95.17 ± 2.05		
FKMI	77.44 ± 3.49	56.98 ± 4.14	71.58 ± 3.54	95.71 ± 1.78		
SVMI	79.35 ± 2.17	57.02 ± 5.75	72.87 ± 3.32	95.68 ± 1.54		
EM	57.45 ± 6.68	55.64 ± 7.36	69.48 ± 3.60	92.35 ± 3.72		
SVDI	57.54 ± 7.01	56.94 ± 5.11	70.82 ± 3.84	93.82 ± 2.93		
BPCA	18.38 ± 0.88	59.74 ± 4.30	64.85 ± 0.32	65.37 ± 1.96		
MC	77.19 ± 2.19	56.74 ± 5.79	71.62 ± 3.15	95.31 ± 1.72		
CMC	**79.62 ± 1.74**	57.12 ± 5.41	71.82 ± 4.10	95.73 ± 1.53		
DNI	64.04 ± 3.00	56.90 ± 5.47	72.60 ± 3.92	95.11 ± 1.87		

EC has been capable of offering an outstanding performance in comparison with the other imputation models.

3.3. Statistical and graphical analysis of the EventCovering method

We have proceeded to a statistical analysis in order to establish the significance degree of differences in performance between the *EC* method and the other imputation methods. We have applied a non-parametric statistical test (Dĕmsar, 2006; García & Herrera, 2008), the Wilcoxon Signed Rank Test. We distinguish between the results obtained by the data sets with natural MVs and the data sets with induced MVs. The obtained results for *EC* versus the other methods can be found in Table 6 for the data sets with natural MVs, and Table 7 for the data sets with induced MVs.

As we can see from Tables 6 and 7, *EC* is the best method in almost all comparisons with statistical significance of $\alpha = 0.05$. There exist some exceptions:

- *IM* has no significant difference with *EC* for the RBFN and RBFNI models in the data sets with natural MVs.
- *SVMI* has no significant difference with *EC* for RBFN in the data sets with natural MVs. It also has a statistical significance with $\alpha = 0.1$ in RBFND method in both natural and artificial MV data sets.
- *MC* presents no significant difference with *EC* in the RBFN method for the data sets with natural MVs. For the artificial data sets, *MC* has statistical significance with $\alpha = 0.1$ in the RBFND method.

Table 6
Wilcoxon signed rank test results for EC (natural MVs).

	RBFN			RBFN decremental			RBFN incremental		
	R−	R+	*p*-value	R−	R+	*p*-value	R−	R+	*p*-value
IM	57	21	0.158	66	12	0.034	59.5	18.5	0.110
KNNI	68	10	0.023	68	10	0.023	71.5	6.5	0.013
WKNNI	70	8	0.015	66	12	0.034	65.5	12.5	0.041
KMI	70	8	0.015	65	13	0.041	67	11	0.028
FKMI	70	8	0.015	68	10	0.023	69.5	8.5	0.022
SVMI	53	25	0.272	61	17	0.084	64.5	13.5	0.050
EM	66	12	0.034	66	12	0.034	67.5	10.5	0.013
SVDI	64	14	0.050	69	9	0.019	71.5	6.5	0.008
BPCA	76	2	0.004	75	3	0.005	67	11	0.028
MC	68	10	0.230	72	6	0.010	68.5	9.5	0.021
CMC	63.5	14.5	0.050	64	14	0.050	68	10.	0.023
DNI	69	9	0.019	66	12	0.034	63.5	12.5	0.041

Table 7
Wilcoxon signed rank test results for EC (artificial MVs).

	RBFN			RBFN decremental			RBFN incremental		
	R−	R+	*p*-value	R−	R+	*p*-value	R−	R+	*p*-value
IM	55	0	0.005	55	0	0.005	54	1	0.007
KNNI	48	7	0.037	43	12	0.114	53	2	0.009
WKNNI	48	7	0.037	49	6	0.028	52	3	0.013
KMI	49	6	0.028	48	7	0.037	52	3	0.013
FKMI	50	5	0.022	51	4	0.017	52	3	0.013
SVMI	48.5	6.5	0.038	46	9	0.059	50	5	0.022
EM	55	0	0.005	52	3	0.013	54	1	0.007
SVDI	55	0	0.005	51	4	0.017	55	0	0.005
BPCA	55	0	0.005	54	1	0.007	51	4	0.017
MC	48	7	0.037	46	9	0.059	52	3	0.013
CMC	48.5	6.5	0.038	44	11	0.093	48	7	0.037
DNI	55	0	0.005	52	3	0.013	54	1	0.007

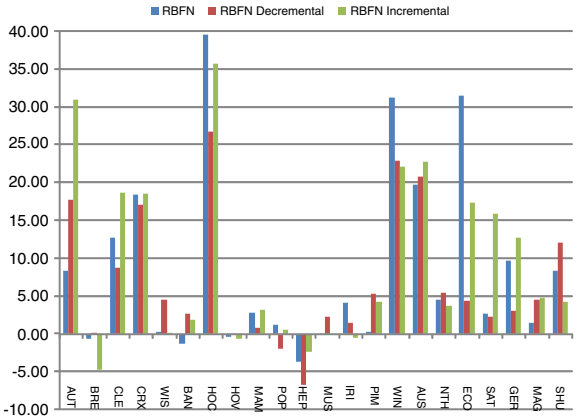

Fig. 1. EC vs. IM.

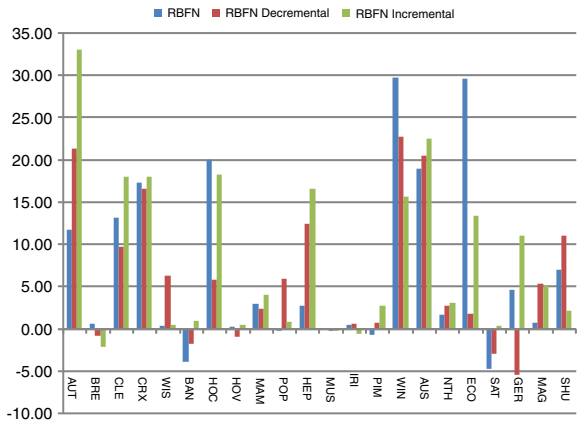

Fig. 2. EC vs. KNNI.

- Finally, *CMC* has a statistical significance with $\alpha = 0.1$ for the artificial data sets in the RBFND method.

These results confirm the previous analysis obtained from the result tables and figures, *EC* method is the best among all presented imputation methods in most cases for the RBFN models considered.

Finally we show graphically that *EC* method has got an outstanding performance with RBFNs. In Figs. 1–12 we use the test set accuracy as a performance measure, and the vertical bars represent the difference in accuracy between *EC* and the other methods for each data set.

- The positive values (bars above the baseline) represent the advantage of *EC* method. That is, the larger the bar, the higher the difference between *EC* method and the indicated one.
- A negative bar under the baseline indicates that the RBFN model for *EC* method has lower accuracy than the other imputation method.

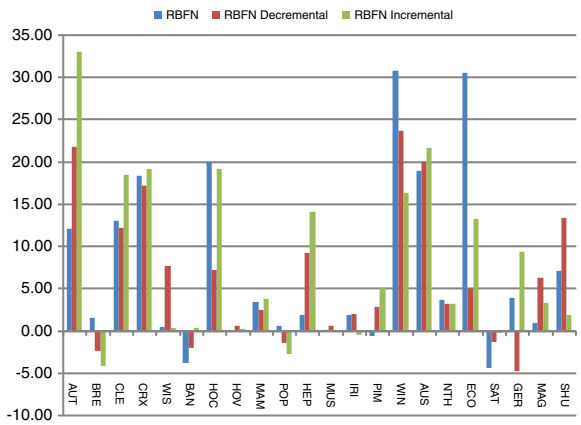

Fig. 3. EC vs. WKKNI.

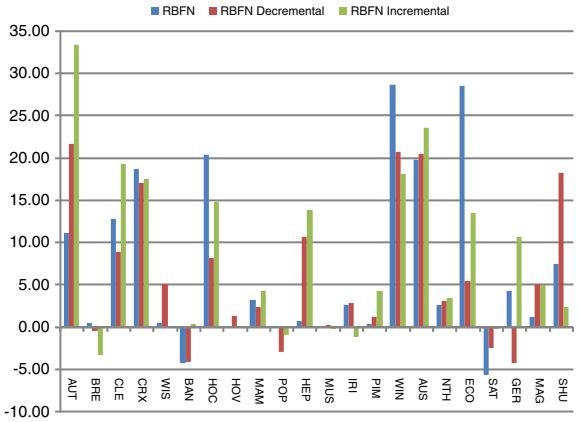

Fig. 4. EC vs. KMI.

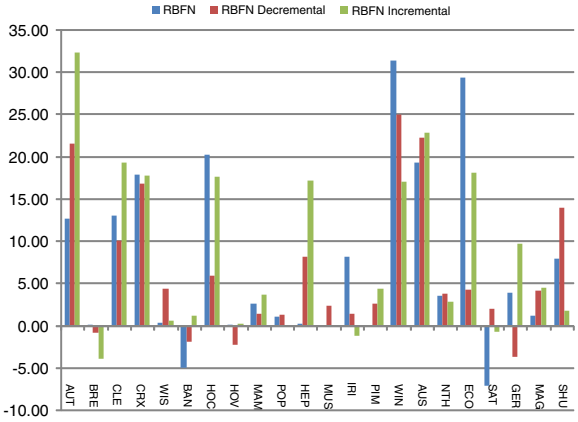

Fig. 5. EC vs. FKMI.

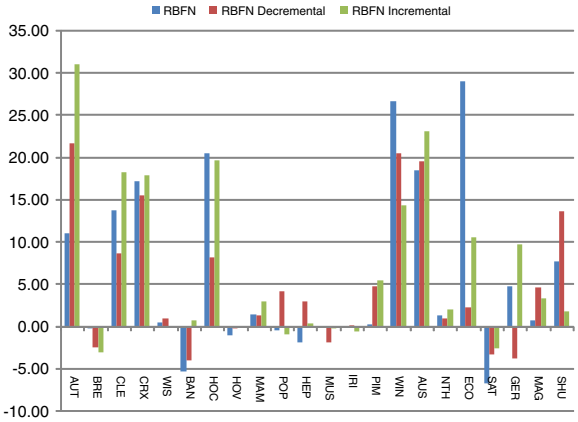

Fig. 6. EC vs. SVMI.

The best behavior of *EC* over all methods is present for almost every data set, since there are more "positive" bars, and with higher positive value. We can distinguish three different situations:

- The data sets AUT, CLE, CRX, HOC, HEP, WIN, AUS, ECO, AUS, GER and SHU present positive differences over the 5% for at least two models of the RBFNs.
- The data sets WIS, HAM, POP, MUS, IRI, PIM, NTH and MAG present little positive difference for *EC*, under the 5%, for at least two RBFN methods.
- Only the data sets BRE, BAN and SAT show a lower accuracy for the RBFN models in the case of *EC* method respect to the rest with a negative bar.

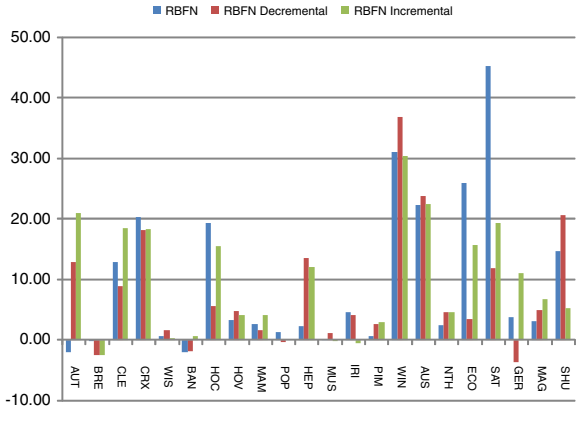

Fig. 7. EC vs. EM.

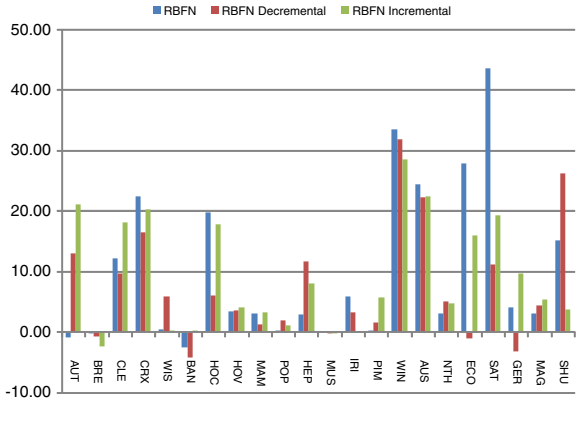

Fig. 8. EC vs. SVDI.

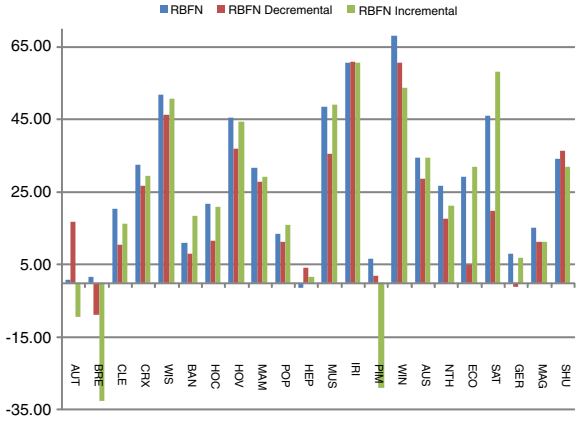

Fig. 9. EC vs. BPCA.

The behavior of the RBFN models in the different data sets is homogeneous, and we can find that the *EC* method outperforms the rest in the same data sets for all the three RBFN models. Therefore, from the figures we can observe that only for the 14% of the data sets, the *EC* imputation method is performing worse than the rest. From the remaining 86%, on the 36% of the data sets the *EC* method behaves better than the rest with little difference, and in the remaining 50% the improvements of accuracy are notorious.

4. Concluding remarks

We have studied the use of imputation techniques for the analysis of RBFN in classification problems, presenting a comparison

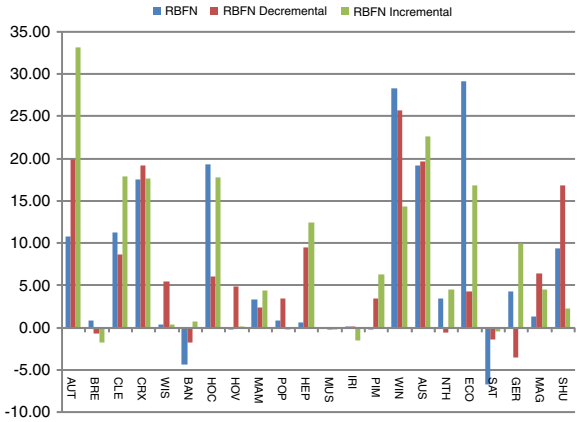

Fig. 10. EC vs. MC.

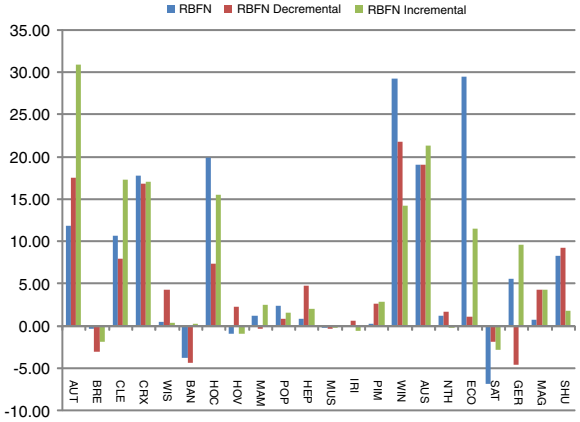

Fig. 11. EC vs. CMC.

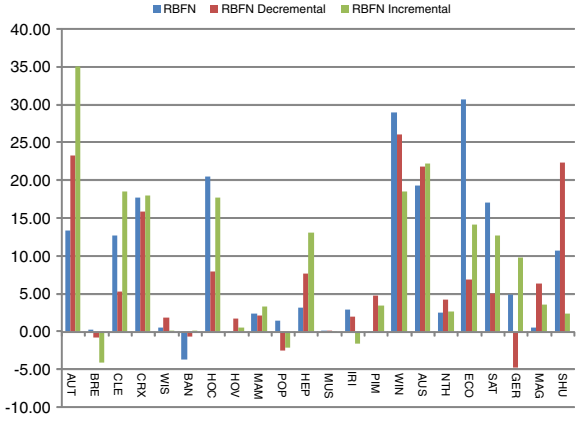

Fig. 12. EC vs. DNI.

between: (1)imputation, (2)do not impute, and (3)ignore cases with MVs.

When comparing the results of these three options, the need for using imputation methods is clear when analyzing RBFN for classification, since improvements in performance of the results are often achieved.

We must point out that the *EC* method is the best choice for carrying out an imputation of MVs when working with RBFNs in three considered variants of this model. Therefore, we can confirm the good synergy between RBFN models and the EventCovering method.

References

Acuna, E., & Rodriguez, C. (2004). The treatment of missing values and its effect in the classifier accuracy. In D. Banks, L. House, F. R. McMorris, P. Arabie, & W. Gaul (Eds.), *Classification, clustering and data mining applications* (pp. 639–648). Berlin, Germany: Springer-Verlag Berlin-Heidelberg.

Arenas-Garcia, J., Gomez-Verdejo, V., & Figueiras-Vidal, A. R. (2007). Fast evaluation of neural networks via confidence rating. *Neurocomputing*, *70*, 2775–2782.

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository [WWW Page]. Irvine, CA: University of California, School of Information and Computer Science. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Batista, G. E. A. P. A., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, *17*, 519–533.

Billings, S. A., Wei, H.-L., & Balikhin, M. A. (2007). Generalized multiscale radial basis function networks. *Neural Networks*, *20*(10), 1081–1094.

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2*, 321–355.

Buhmann, M. D. (2003). *Cambridge monographs on applied and computational mathematics. Radial basis functions: Theory and implementations*.

Děmsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Eickhoff, R., & Ruckert, U. (2007). Robustness of radial basis functions. *Neurocomputing*, *70*, 2758–2767.

Ennett, C. M., Frize, M., & Walker, C. R. (2001). Influence of missing values on artificial neural network performance. *Medinfo*, *10*, 449–453.

Er, M. J., Wu, S., Lu, J., & Hock-Lye, T. (2002). Face recognition with radial basis function (RBF) neural networks. *IEEE Transactions on Neural Networks*, *13*, 697–710.

Farhangfar, A., Kurgan, L., & Pedrycz, W. (2004). Experimental analysis of methods for imputation of missing values in databases. In K. L. Priddy (Ed.). *SPIE. Vol. 5421. Intelligent computing: Theory and applications II*. Michigan (pp. 172–182).

Feng, H. A. B., Chen, G. C., Yin, C. D., Yang, B. B., & Chen, Y. E. (2005). A SVM regression based approach to filling in missing values. In R. Khosla, R. J. Howlett, & L. C. Jain (Eds.), *Lecture notes in artificial intelligence*: *Vol. 3683. Knowledge-based intelligent information and engineering systems (KES 2005)* (pp. 581–587). Springer.

García, S., & Herrera, F. (2008). An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677–2694.

Ghodsi, A., & Schuurmans, D. (2003). Automatic basis selection techniques for RBF networks. *Neural Networks*, *16*(5–6), 809–816.

Grzymala-Busse, J. W., & Hu, M. (2000). A comparison of several approaches to missing attribute values in data mining. In W. Ziarko, & Y. Y. Yao (Eds.), *Lecture notes in computer science*: *Vol. 2005. Rough sets and current trends in computing : Second international conference (RSCTC 2000)* (pp. 378–385). Canada: Springer.

Grzymala-Busse, J. W., & Goodwin, L. K. (2005). Handling missing attribute values in preterm birth data sets. In D. Slezak, J. Yao, J. F. Peters, W. Ziarko, & X. Hu (Eds.), *Lecture notes in computer science*: *Vol. 3642. Rough sets, fuzzy sets, data mining, and granular computing (RSFDGrC 2005)* (pp. 342–351). Canada: Springer.

Harpham, C., & Dawson, C. W. (2006). The effect of different basis functions on a radial basis function network for time series prediction: A comparative study. *Neurocomputing*, *69*, 2161–2170.

Kros, J. F., Lin, M., & Brown, M. L. (2006). Effects of the neural network s-Sigmoid function on KDD in the presence of imprecise data. *Computers and Operations Research*, *33*, 3136–3149.

Lázaro, M., Santamaría, I., & Pantaleón, C. (2003). A new EM-based training algorithm for RBF networks. *Neural Networks*, *16*(1), 69–77.

Lendasse, A., Francois, D., Wertz, V., & Verleysen, M. (2005). Vector quantization: A weighted version for time-series forecasting. *Future Generation Computer Systems*, *21*, 1056–1067.

Li, D., Deogun, J., Spaulding, W., & Shuart, B. (2004). Towards missing data imputation: A study of fuzzy K-means clustering method. In S Tsumoto, R. Slowinski, J. Komorowski, & J. W. Grzymala-Busse (Eds.), *Lecture notes in computer science*: *Vol. 3066. Rough sets and current trends in computing (RSCTC 2004)* (pp. 573–579). Sweden: Springer-Verlag.

Liao, Y., Fang, S.-C., & Nuttle, H. L. W. (2003). Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Networks*, *16*(7), 1019–1028.

Lim, C. P., Leong, J. H., & Kuan, M. M. (2005). A hybrid neural network system for pattern classification tasks with missing features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 648–653.

Little, R. J., & Rubin, D. B. (1987). *Statistical analysis with missing data*. New York: John Wiley and Sons.

Markey, M. K., Tourassi, G. D., Margolis, M., & DeLong, D. M. (2006). Impact of missing data in evaluating artificial neural networks trained on complete data. *Computers in Biology and Medicine*, *36*, 516–525.

Mileva-Boshkoska, B., & Stankovski, M. (2007). Prediction of missing data for ozone concentrations using support vector machines and radial basis neural networks. *Informatica (Ljubljana)*, *31*, 425–430.

Morris, C. W., Boddy, L., & Wilkins, M. F. (2001). Effects of missing data on RBF neural network identification of biological taxa: Discrimination of microalgae from flow cytometry data. *International Journal of Smart Engineering System Design*, *3*, 195–202.

Musavi, M. T., Ahmed, W., Chan, K. H., Faris, K. B., & Hummels, D. M. (1992). On the training of radial basis function classifiers. *Neural Networks*, *5*, 595–603.

Nelwamondo, F. V., Mohamed, S., & Marwala, T. (2007). Missing data: A comparison of neural network and expectation maximization techniques. *Current Science*, *93*, 1514–1521.
Oba, S., Sato, M., Takemasa, I., Monden, M., Matsubara, K., & Ishii, S. (2003). A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, *19*, 2088–2096.
Pearson, R. K. (2005). *Mining imperfect data*. SIAM.
Pelckmans, K., De Brabanterb, J., Suykensa, J. A. K., & De Moor, B. (2005). Handling missing values in support vector machine classifiers. *Neural Networks*, *18*, 684–692.
Pisoni, E., Pastor, F., & Volta, M. (2008). Artificial neural networks to reconstruct incomplete satellite data: Application to the mediterranean sea surface temperature. *Nonlinear Processes in Geophysics*, *15*, 61–70.
Plat, J. (1991). A resource allocating network for function interpolation. *Neural Computation*, *3*, 213–225.
Powell, M. J. D. (1987). Radial basis function for multivariate interpolation: A review. In J. C. Mason, & M. G. Cox (Eds.), *Algorithm for approximation* (pp. 143–168). Oxford, England: Clarendon Press.
Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann Publishers.
Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychol Methods*, *7*, 147–177.
Schneider, T. (2001). Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, *14*, 853–871.
Schwenker, F., Kestler, H. A., & Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural Networks*, *14*(4–5), 439–458.
Sun, Y. V., & Kardia, S. L. R. (2008). Imputing missing genotypic data of single-nucleotide polymorphisms using neural networks. *European Journal of Human Genetics*, *16*, 487–495.
Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., & Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, *17*, 520–525.
Uysal, M. (2007). Reconstruction of time series data with missing values. *Journal of Applied Sciences*, *7*, 922–925.
Wallace, M., Tsapatsoulis, N., & Kollias, S. (2005). Intelligent initialization of resource allocating RBF networks. *Neural Networks*, *18*(2), 117–122.
Wang, S. (2003). Application of self-organising maps for data mining with incomplete data sets. *Neural Computation & Applications*, *12*, 42–48.
Wei, H., & Amari, S.-i. (2008). Dynamics of learning near singularities in radial basis function networks. *Neural Networks*, *21*(7), 989–1005.
Wong, A. K. C., & Chiu, D. K. Y. (1987). Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *9*, 796–805.
Yeung, D. S., Ng, W. W. Y., Wang, D., Tsang, E. C. C., & Wang, X.-Z. (2007). Localized generalization error model and its application to architecture selection for radial basis function neural network. *IEEE Transactions on Neural Networks*, *18*, 1294–1305.
Yingwei, L., Sundararajan, N., & Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, *9*, 361–478.
Yoon, S. Y., & Lee, S. Y. (1999). Training algorithm with incomplete data for feed-forward neural networks. *Neural Processing Letters*, *10*, 171–179.

1.2. On the choice of an imputation method for missing values. A study of three groups of classification methods: rule induction learning, lazy learning and approximate methods

- J. Luengo, S. García, F. Herrera, On the choice of an imputation method for missing values. A study of three groups of classification methods: rule induction learning, lazy learning and approximate methods. **Submitted to Knowledge and Information Systems**.
  - Status: **Submitted**.
  - Impact Factor (JCR 2009): 2.211.
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 27 / 103.
  - Subject Category: Computer Science, Information Systems. Ranking 24 / 116.

Editorial Manager(tm) for Knowledge and Information Systems
Manuscript Draft

Manuscript Number: KAIS-2083R2

Title: On the choice of an imputation method for missing values. A study of three groups of classification methods: rule induction learning, lazy learning and approximate methods

Article Type: Regular Paper

Keywords: Approximate Models; Classification; Imputation; Rule Induction Learning; Lazy Learning; Missing Values; Single Imputation

Corresponding Author: Mr. Julian Luengo,

Corresponding Author's Institution: University of Granada

First Author: Julian Luengo

Order of Authors: Julian Luengo; Salvador García, Dr.; Francisco Herrera, Dr.

Abstract: In real-life data, information is frequently lost in data mining, caused by the presence of missing values in attributes. Several schemes have been studied to overcome the drawbacks produced by missing values in data mining tasks; one of the most well known is based on preprocessing, formerly known as imputation. In this work we focus on a classification task with twenty-three classification methods and fourteen different approaches to Missing attribute Values treatment that are presented and analyzed. The analysis involves a group-based approach, in which we distinguish between three different categories of classification methods.
Each category behaves differently, and the evidence obtained shows that the use of determined Missing Values imputation methods could improve the accuracy obtained for these methods.

In this study, the convenience of using imputation methods for pre-processing data sets with Missing Values is stated. The analysis suggests that
the use of particular imputation methods conditioned to the groups is required.

On the choice of an imputation method for missing values. A study of three groups of classification methods: rule induction learning, lazy learning and approximate methods

Julián Luengo[1], Salvador García[2], Francisco Herrera[1]
[1]Dept. of Computer Science and Artificial Intelligence,
CITIC-University of Granada, 18071, Granada, Spain.
e-mail: {julianlm,herrera}@decsai.ugr.es
[2]Dept. of Computer Science,
University of Jaén, 23071, Jaén, Spain.
e-mail: sglopez@ujaen.es

Abstract

In real-life data, information is frequently lost in data mining, caused by the presence of missing values in attributes. Several schemes have been studied to overcome the drawbacks produced by missing values in data mining tasks; one of the most well known is based on preprocessing, formerly known as imputation. In this work we focus on a classification task with twenty-three classification methods and fourteen different approaches to Missing attribute Values treatment that are presented and analyzed. The analysis involves a group-based approach, in which we distinguish between three different categories of classification methods. Each category behaves differently, and the evidence obtained shows that the use of determined Missing Values imputation methods could improve the accuracy obtained for these methods.

In this study, the convenience of using imputation methods for preprocessing data sets with Missing Values is stated. The analysis suggests that the use of particular imputation methods conditioned to the groups is required.

Keywords: Approximate Models; Classification; Imputation; Rule Induction Learning; Lazy Learning; Missing Values; Single Imputation

1 Introduction

Many existing, industrial and research data sets contain Missing Values (MVs). There are various reasons for their existence, such as manual data entry procedures, equipment errors and incorrect measurements. The presence of such

imperfections requires a preprocessing stage in which the data is prepared and cleaned (Pyle, 1999), in order to be useful to and sufficiently clear for the knowledge extraction process. The simplest way of dealing with missing values is to discard the examples that contain them. However, this method is practical only when the data contains a relatively small number of examples with MVs and when analysis of the complete examples will not lead to serious bias during the inference (Little and Rubin, 1987).

MVs make the performance of data analysis difficult. The presence of missing values can also pose serious problems for researchers. In fact, inappropriate handling of missing data in the analysis may introduce bias and can result in misleading conclusions being drawn from a research study, and can also limit the generalizability of the research findings (Wang and Wang, 2010). Three types of problem are usually associated with missing values in data mining (Barnard and Meng, 1999): 1) loss of efficiency; 2) complications in handling and analyzing the data; and 3) bias resulting from differences between missing and complete data.

In the particular case of classification, learning from incomplete data becomes even more important. Incomplete data in either the training set or test set or in both sets affect the prediction accuracy of learned classifiers (Gheyas and Smith, 2010). The seriousness of this problem depends in part on the proportion of missing data. Most classification algorithms cannot work directly with incomplete data sets and due to the high dimensionality of real problems it is possible that no valid (complete) cases would be present in the data set (García-Laencina, Sancho-Gómez and Figueiras-Vidal, 2009). Therefore, it is important to analyze which is the best technique or preprocessing considered in order to treat the present MVs before applying the classification methods as no other option is possible.

Usually the treatment of missing data in data mining can be handled in three different ways (Farhangfar, Kurgan and Pedrycz, 2007):

- The first approach is to discard the examples with missing data in their attributes. Therefore deleting attributes with elevated levels of missing data is included in this category too.
- Another approach is the use of maximum likelihood procedures, where the parameters of a model for the complete data are estimated, and later used for imputation by means of sampling.
- Finally, the imputation of MVs is a class of procedures that aims to fill in the MVs with estimated ones. In most cases, a data set's attributes are not independent from each other. Thus, through the identification of relationships among attributes, MVs can be determined

We will focus our attention on the use of imputation methods. A fundamental advantage of this approach is that the missing data treatment is independent of the learning algorithm used. For this reason, the user can select the most appropriate method for each situation he faces. There is a wide family of imputation methods, from simple imputation techniques like mean substitution,

K-Nearest Neighbour, etc.; to those which analyze the relationships between attributes such as: support vector machines-based, clustering-based, logistic regressions, maximum-likelihood procedures and multiple imputation (Batista and Monard, 2003; Farhangfar, Kurgan and Dy, 2008).

The literature on imputation methods in Data Mining employs well-known Machine Learning methods for their studies, in which the authors show the convenience of imputing the MVs for the mentioned algorithms, particularly for classification. The vast majority of MVs studies in classification usually analyze and compare one imputation method against a few others under controlled amounts of MVs, and induce them artificially with known mechanisms and probability distributions (Acuna and Rodriguez, 2004; Batista and Monard, 2003; Farhangfar et al., 2008; Hruschka, Hruschka and Ebecken, 2007; Li, Deogun, Spaulding and Shuart, 2004).

We want to analyze the effect of the use of a large set of imputation methods on all the considered classifiers. Most of the considered classification methods have been used previously in MVs studies. However, they have been considered all together. In this work we will establish three groups of classifiers to categorize them, and we will examine the best imputation strategies for each group. The former groups are as follows:

- The first group consists of the *Rule Induction Learning* category. This group refers to algorithms which infer rules using different strategies. Therefore, we can identify as belonging to this category those methods that produce a set of more or less interpretable rules. These rules include discrete and/or continuous features, which will be treated by each method depending on their definition and representation.

- The second group represents the *Approximate Models*. It includes Artificial Neural Networks, Support Vector Machines and Statistical Learning. In this group we include the methods which act like a black box. Therefore, those methods which do not produce an interpretable model fall under this category. Although the Naïve Bayes method is not a completely black box method, we have considered that this is the most appropriate category for it.

- The third and last group corresponds to the *Lazy Learning* category. This group includes methods which do not create any model, but use the training data to perform the classification directly. This process implies the presence of measures of similarity of some kind. Thus, the methods which use a similarity function to relate the inputs to the training set are considered as belonging to this category.

In order to perform the analysis, we use a large bunch of data sets, twenty-one in total, with natural MVs. All the data sets have their proper MVs and we do not induce them, as we want to stay as close to the real world data as possible. First, we analyze the use of the different imputation strategies versus case deletion and the total lack of missing data treatment, for a total of fourteen imputation methods. Therefore, each one of the twenty-three imputation methods

is used over the fourteen imputation results. All the imputation and classification algorithms are publicly available in the KEEL software[1] (Alcalá-fdez, Sánchez, García, Jesus, Ventura, Garrell, Otero, Bacardit, Rivas, Fernández and Herrera, 2009). These results are compared using the Wilcoxon Signed Rank test (Demšar, 2006; García and Herrera, 2008) in order to obtain the best method(s) for each classifier. With this information we can extract the best imputation method for the three groups, and indicate the best global option using a set of average rankings.

We have also analyzed two metrics related to the data characteristics, formerly known as Wilson's noise ratio and Mutual Information. Using these measures, we have observed the influence of the imputation procedures on the noise and on the relationship of the attributes with the class label as well. This procedure tries to quantify the quality of each imputation method independently of the classification algorithm.

The obtained results will help us to explain how imputation may be a useful tool to overcome the negative impact of missing data, and the most suitable imputation method for each classifier, each group and all the classification methods together.

The rest of the paper is organized as follows. In Section 2 we present the basis of the application of the imputation methods, the description of the imputation methods we have used and a brief review of the current state of the art in imputation methods. In Section 3, the experimental framework, the classification methods and the parameters used for both imputation and classification methods are presented. In Section 4, the results obtained are analyzed. In Section 5 we use two measures to quantify the influence of the imputation methods in the data sets, both in the instances and in the features. Finally, in Section 6 we make some concluding remarks.

2 Imputation background

In this section we first set the basis of our study in accordance with the MV literature. The rest of this section is organized as follows: In Subsection 2.1 we have summarized the imputation methods that we have used in our study; in Subsection 2.2 we show a brief snapshot of the latest advances in imputation methods.

A more extensive and detailed description of these methods can be found on the web page `http://sci2s.ugr.es/MVDM`, and a PDF file with the original source paper descriptions is present on the web page formerly named "Imputation of Missing Values. Methods' Description". A more complete bibliography section is also available on the mentioned web page.

It is important to categorize the mechanisms which lead to the introduction of MVs (Little and Rubin, 1987). The assumptions we make about the missingness mechanism and the missing data pattern of missing values can affect which

[1] `http://keel.es`

imputation method could be applied, if any. As Little and Rubin (1987) stated, there are three different mechanisms for missing data induction.

1. Missing completely at random (**MCAR**), when the distribution of an example having a missing value for an attribute does not depend on either the observed data or the missing data.

2. Missing at random (**MAR**), when the distribution of an example having a missing value for an attribute depends on the observed data, but does not depend on the missing data.

3. Not missing at random (**NMAR**), when the distribution of an example having a missing value for an attribute depends on the missing values.

In the case of the MCAR mode, the assumption is that the underlying distributions of missing and complete data are the same, while for the MAR mode they are different, and the missing data can be predicted by using the complete data (Little and Rubin, 1987). These two mechanisms are assumed by the imputation methods so far. As Farhangfar et al. (2008) and Matsubara, Prati, Batista and Monard (2008) state, it is only in the MCAR mechanism case where the analysis of the remaining complete data (ignoring the incomplete data) could give a valid inference (classification in our case) due to the assumption of equal distributions. That is, case and attribute removal with missing data should be applied only if the missing data is MCAR, as both of the other mechanisms could potentially lead to information loss that would lead to the generation of a biased/incorrect classifier (i.e. a classifier based on a different distribution).

Another approach is to convert the missing values to a new value (encode them into a new numerical value), but such a simplistic method was shown to lead to serious inference problems (Schafer, 1997). On the other hand, if a significant number of examples contain missing values for a relatively small number of attributes, it may be beneficial to perform imputation (filling-in) of the missing values. In order to do so, the assumption of MAR randomness is needed, as Little and Rubin (1987) observed in their analysis.

In our case we will use single imputation methods, due to the time complexity of the multiple imputation schemes, and the assumptions they make regarding data distribution and MV randomness; that is, that we should know the underlying distributions of the complete data and missing data prior to their application.

2.1 Description of the imputation methods

In this subsection, we briefly describe the imputation methods that we have used.

- Do Not Impute (**DNI**). As its name indicates, all the missing data remains unreplaced, so the networks must use their default MVs strategies. The objective is to verify whether imputation methods allow the classification methods to perform better than when using the original data sets.

As a guideline, in Grzymala-Busse and Hu (2000) a previous study of imputation methods is presented.

- Case deletion or Ignore Missing (**IM**). Using this method, all instances with at least one MV are discarded from the data set.

- Global Most Common Attribute Value for Symbolic Attributes, and Global Average Value for Numerical Attributes (**MC**)(Grzymala-Busse, Goodwin, Grzymala-Busse and Zheng, 2005). This method is very simple: for nominal attributes, the MV is replaced with the most common attribute value, and numerical values are replaced with the average of all values of the corresponding attribute.

- Concept Most Common Attribute Value for Symbolic Attributes, and Concept Average Value for Numerical Attributes (**CMC**)(Grzymala-Busse et al., 2005). As stated in *MC*, the MV is replaced by the most repeated one if nominal or the mean value if numerical, but considering only the instances with the same class as the reference instance.

- Imputation with K-Nearest Neighbor (**KNNI**)(Batista and Monard, 2003). Using this instance-based algorithm, every time an MV is found in a current instance, KNNI computes the k nearest neighbors and a value from them is imputed. For nominal values, the most common value among all neighbors is taken, and for numerical values the average value is used. Therefore, a proximity measure between instances is needed for it to be defined. The euclidean distance (it is a case of a L_p norm distance) is the most commonly used in the literature.

- Weighted imputation with K-Nearest Neighbor (**WKNNI**)(Troyanskaya, Cantor, Sherlock, Brown, Hastie, Tibshirani, Botstein and Altman, 2001). The Weighted K-Nearest Neighbor method selects the instances with similar values (in terms of distance) to a considered one, so it can impute as *KNNI* does. However, the estimated value now takes into account the different distances from the neighbors, using a weighted mean or the most repeated value according to the distance.

- K-means Clustering Imputation (**KMI**)(Li et al., 2004). Given a set of objects, the overall objective of clustering is to divide the data set into groups based on the similarity of objects, and to minimize the intra-cluster dissimilarity. KMI measures the intra-cluster dissimilarity by the addition of distances among the objects and the centroid of the cluster which they are assigned to. A cluster centroid represents the mean value of the objects in the cluster. Once the clusters have converged, the last process is to fill in all the non-reference attributes for each incomplete object based on the cluster information. Data objects that belong to the same cluster are taken to be nearest neighbors of each other, and KMI applies a nearest neighbor algorithm to replace missing data, in a similar way to KNNI.

- Imputation with Fuzzy K-means Clustering (**FKMI**)(Acuna and Rodriguez, 2004; Li et al., 2004). In fuzzy clustering, each data object has a membership function which describes the degree to which this data object belongs to a certain cluster. In the process of updating membership functions and centroids, FKMI's only take into account complete attributes. In this process, the data object cannot be assigned to a concrete cluster represented by a cluster centroid (as is done in the basic K-mean clustering algorithm), because each data object belongs to all K clusters with different membership degrees. FKMI replaces non-reference attributes for each incomplete data object based on the information about membership degrees and the values of cluster centroids.

- Support Vector Machines Imputation (**SVMI**)(Feng, Guoshun, Cheng, Yang and Chen, 2005) is an SVM regression based algorithm to fill in missing data, i.e. set the decision attributes (output or classes) as the condition attributes (input attributes) and the condition attributes as the decision attributes, so SVM regression can be used to predict the missing condition attribute values. In order to do that, first SVMI selects the examples in which there are no missing attribute values. In the next step the method sets one of the condition attributes (input attribute), some of those values that are missing, as the decision attribute (output attribute), and the decision attributes as the condition attributes by contraries. Finally, an SVM regression is used to predict the decision attribute values.

- Event Covering (**EC**)(Wong and Chiu, 1987). Based on the work of Wong et al., a mixed-mode probability model is approximated by a discrete one. First, EC discretizes the continuous components using a minimum loss of information criterion. Treating a mixed-mode feature n-tuple as a discrete-valued one, a new statistical approach is proposed for the synthesis of knowledge based on cluster analysis. The main advantage of this method is that it does not require either scale normalization or the ordering of discrete values. By synthesizing the data into statistical knowledge, the EC method involves the following processes: 1) synthesize and detect from data inherent patterns which indicate statistical interdependency; 2) group the given data into inherent clusters based on this detected interdependency; and 3) interpret the underlying patterns for each cluster identified. The method of synthesis is based on the author's *event–covering* approach. With the developed inference method, EC is able to estimate the MVs in the data.

- Regularized Expectation-Maximization (**EM**)(Schneider, 2001). Missing values are imputed with a regularized expectation maximization (EM) algorithm. In an iteration of the EM algorithm, given estimates of the mean and of the covariance matrix are revised in three steps. First, for each record with missing values, the regression parameters of the variables with missing values among the variables with available values are computed from the estimates of the mean and of the covariance matrix.

Second, the missing values in a record are filled in with their conditional expectation values given the available values and the estimates of the mean and of the covariance matrix, the conditional expectation values being the product of the available values and the estimated regression coefficients. Third, the mean and the covariance matrix are re-estimated, the mean as the sample mean of the completed data set and the covariance matrix as the sum of the sample covariance matrix of the completed data set and an estimate of the conditional covariance matrix of the imputation error. The EM algorithm starts with initial estimates of the mean and of the covariance matrix and cycles through these steps until the imputed values and the estimates of the mean and of the covariance matrix stop changing appreciably from one iteration to the next.

- Singular Value Decomposition Imputation (**SVDI**)(Troyanskaya et al., 2001). In this method, singular value decomposition is used to obtain a set of mutually orthogonal expression patterns that can be linearly combined to approximate the values of all attributes in the data set. In order to do that, first SVDI estimates the MVs within the *EM* algorithm, and then it computes the Singular Value Decomposition and obtains the eigenvalues. Now SVDI can use the eigenvalues to apply a regression to the complete attributes of the instance, to obtain an estimation of the MV itself.

- Bayesian Principal Component Analysis(**BPCA**)(Oba, aki Sato, Takemasa, Monden, ichi Matsubara and Ishii, 2003). This method is an estimation method for missing values, which is based on Bayesian principal component analysis. Although the methodology that a probabilistic model and latent variables are estimated simultaneously within the framework of Bayesian inference is not new in principle, actual BPCA implementation that makes it possible to estimate arbitrary missing variables is new in terms of statistical methodology. The missing value estimation method based on BPCA consists of three elementary processes. They are (1) principal component (PC) regression, (2) Bayesian estimation, and (3) an expectationmaximization (EM)-like repetitive algorithm.

- Local Least Squares Imputation (**LLSI**)(Kim, Golub and Park, 2005). With this method, a target instance that has missing values is represented as a linear combination of similar instances. Rather than using all available genes in the data, only similar genes based on a similarity measure are used. The method has the "local" connotation. There are two steps in the LLSI. The first step is to select k genes by the L_2-norm. The second step is regression and estimation, regardless of how the k genes are selected. A heuristic k parameter selection method is used by the authors.

2.2 An overview of the analysis of imputation methods in the literature for classification

This section tries to present a snapshot of the general studies of analysis of MVs that we can find in the literature for classification involving the imputation procedure.

- Grzymala-Busse and Hu (2000) compares the performance of the LERS classification method with the application of nine different methods for MVs: the C4.5 probability-based mechanism, MC, CMC, LEM2 based, EC and assigns all possible values. Their results state that the use of imputation methods show that, on average, imputation helps to improve classification accuracy, and the best imputation for LERS was achieved with the C4.5 internal method.

- Batista and Monard (2003) tested the classification accuracy of two popular classifiers (C4.5 and CN2) and two imputation methods: KNNI and MC. Both CN2 and C4.5 (like (Grzymala-Busse and Hu, 2000)) algorithms have their own MV estimation. From their study, KNNI results in good accuracy, but only when the attributes are not highly correlated to each other.

- Acuna and Rodriguez (2004) have investigated the effect of four methods that deal with MVs. As in (Batista and Monard, 2003), they use KNNI and two other imputation methods (MC and median imputation). They also use the K-NN and Linear Discriminant Analysis classifiers. The results of their study show that no significant harmful effect in accuracy is obtained from the imputation procedure. In addition to this, they state that the KNNI method is more robust in the increment of MVs in the data set in respect to the other compared methods.

- Li et al. (2004) apply Soft Computing to MVs using a fuzzy clustering method. They compare the FKMI with Mean substitution and KMI. Using a Root Mean Square Error error analysis, they state that the basic KMI algorithm outperforms the MC method. Experiments also show that the overall performance of the FKMI method is better than the basic KMI method, particularly when the percentage of missing values is high.

- Feng et al. (2005) use the SVMI for filling in MVs. They do not compare this with any other imputation methods. Furthermore, they state that we should select enough complete examples where there is no missing data as the training data set.

- Nogueira, Santos and Zárate (2007) presented a comparison of techniques used to recover values in a real imbalanced database, with a massive occurrence of missing data. This makes the process of obtaining a set of representative records, used for the recovering techniques, difficult. They used C4.5, Naïve-Bayes, K-NN and Multi-Layer Perceptron as classifiers.

To treat the MVs, they applied several techniques: default value substitution or related attribute recovery. The latter tries to obtain the missing value from the information of another attribute. In addition to this, cleaning of instances/attributes with too many MVs was also carried out.

- Saar-Tsechansky and Provost (2007) compare several different methods (predictive value imputation, the distribution-based imputation used by C4.5 and using reduced models) for applying classification trees to instances with missing values. They distinguish between MVs in "training" (usual MVs), and MVs in "prediction" time (i.e. test partition), and adapt the novel reduced-models to this scenario. The results show that for the predictive value imputation and C4.5 distribution-based, both can be preferable under different conditions. Their novel technique (reduced models) consistently outperforms the other two methods based on their experimentation.

- Hruschka et al. (2007) propose two imputation methods based on Bayesian networks. They compare them with 4 classical imputation methods: EM, Data Augmentation, C4.5 and the CMC method, using 4 nominal data sets from the UCI repository (Asuncion and Newman, 2007) with natural MVs (but inducing MVs in them as well). In their analysis, they employ 4 classifiers: One-Rule, Naïve-Bayes, C4.5 and PART. As performance measures, the authors measure the prediction value (i.e. the similarity of the imputed value to the original removed one) and the classification accuracy obtained with the four mentioned models. Computing times consumed to perform the imputations are also reported. From the results, the authors state that better prediction results do not imply better classification results.

- Farhangfar et al. (2007) take as the objective of their paper to develop a unified framework supporting a host of imputation methods. Their study inserts some imputation methods into their framework (Naïve-Bayes and Hot Deck) and compares this with other basic methods: Mean, Linear Discriminant Analysis, Logreg, etc. All their experimentation is based on discrete data, so they use the "accuracy" of imputed values against randomly generated missing data. The relation of this imputation accuracy to classification accuracy is not studied.

- Farhangfar et al. (2008) perform a comprehensive study on imputation for discrete data sets, comparing with classical imputation methods and the framework they proposed in (Farhangfar et al., 2007). This study uses a representative method of several classifiers' types: Decision Trees, Instance-Based Learning, Rule-Based classifier, Probabilistic methods and SVMs. The missing data is produced artificially in a wide ranging amount for each of the data sets, and the results obtained from the classification of imputed data are compared with the ones on missing data. This study shows that the impact of the imputation varies among different classifiers,

and that imputation is beneficial for most amounts of missing data above 5% and that the amount of improvement does not depend on the amount of missing data. The performed experimental study also shows that there is no universally best imputation method.

- Matsubara et al. (2008) present an adaptation of a semi-supervised learning algorithm for imputation. They impute the MV using the C4.5 and Naïve-Bayes classifiers by means of a ranking aggregation to select the best examples. They compare the method with three qualitative UCI (Asuncion and Newman, 2007) data sets applying artificial MVs, and perform a similar study to the one presented by Batista and Monard (2003), comparing with the KNNI and MC methods. Using a non-parametric statistical test they demonstrate the better performance of the new method over the other two in some cases.

- Song, Shepperd, Chen and Liu (2008) study the relationship between the use of the KNNI method, and the C4.5 performance (counting with its proper MV technique) over 6 data sets of software projects. They emphasize the different MVs' mechanisms (MCAR, MAR and NMAR), and the amount of MVs introduced. From their analysis, they found results which agree with Batista and Monard (2003): KNNI can improve the C4.5 accuracy. They ran a Mann-Whitney statistical test to obtain significant differences in this statement. They also show that the missingness mechanism and pattern affect the classifier and imputation method performance.

- García-Laencina et al. (2009) evaluate the influence of imputing missing values into the classification accuracy obtained by an artificial neural network (multi-layer perceptron). Four imputation techniques are considered: KNNI, SOM imputation, MLP imputation, and EM over one synthetic and two real data sets, varying the amount of MVs introduced. They conclude that in real-life scenarios a detailed study is required in order to evaluate which missing data estimation can help to enhance the classification accuracy.

- Twala (2009) empirically analyzes 7 different procedures to treat artificial MVs for decision trees over 21 real data sets. From the study it can be concluded that listwise deletion is the worst choice, while the multiple imputation strategy performs better than the rest of the imputation methods (particularly those with high amounts of MVs), although there is no outstanding procedure.

- Wang and Wang (2010) take an example of a marketing survey to explain how the complete instances subset of the data set can be used to train a classification method in order to obtain information about the MVs. Using the trained classifier, artificial imputation values are generated. Those artificial values make the artificial complete observations more predictable by the trained classifier than other artificial values do, then this value are more likely to be true than the others.

- Ding and Simonoff (2010) investigate eight different missingness patterns, depending on the relationship between the missingness and three types of variables, the observed predictors, the unobserved predictors (the missing values) and the response variable. They focus on the case of classification trees for binary data (C4.5 and CART) using a modeling bankruptcy database, showing that the relationship between the missingness and the dependent variable, as well as the existence or non-existence of missing values in the testing data, are the most helpful criteria to distinguish different missing data methods.

- Merlin, Sorjamaa, Maillet and Lendasse (2010) propose a a new method for the determination of missing values in temporal databases based on self-organizing maps. Using two classifiers for the spatial and temporal dependencies, improvements in respect to the EM method in a hedge fund problem are shown.

- Luengo, García and Herrera (2010) study several imputation methods for RBFNs classifiers, both for natural and artificial (MCAR) MVs. From their results can be seen that the EventCovering method has a good synergy with respect to the RBFN methods, as it provides better improvements in classification accuracy.

- Gheyas and Smith (2010) propose a single imputation method and a multiple imputation method, both of them based on a generalized regression neural network (GRNN). Their proposal is compared with 25 imputation methods of different natures, from Machine Learning methods to several variants of GRNNs. 98 data sets are used in order to introduce MVs with MCAR, MAR and NMAR mechanisms. Then the results of the imputation methods are compared by means of 3 different criteria, using three classifiers: MLP, logistic regression and a GRNN-based classifier, showing the advantages of the proposal.

As we can appreciate from the mentioned studies they present heterogeneity. There are many different approaches to the treatment of MVs, which use many different methods (to classify and to impute MVs), but they produce similar conclusions about the convenience of using imputation methods. Therefore, in spite of the variety of studies presented, the necessity of using imputation methods for MVs is demonstrated.

3 Experimental framework

When analyzing imputation methods, a wide range of set ups can be observed. The data sets used, their type (real or synthetic), the origin and amount of MVs, etc. must be carefully described, as the results will strongly depend on them. All these aspects are described in Subsection 3.1.

The good or bad estimations performed by the imputation method will be analyzed with regard to the accuracy obtained by many classification methods.

They are presented in Subsection 3.2, grouped in the different families that we have considered, so that we can extract specialized conclusions relating the imputation results to similar classification methods.

Not all the classification methods are capable of managing the MVs on their own. It is important to indicate which methods can, and the strategy that we follow when the contrary case occurs. In Subsection 3.3 we tackle the different situations which appear when using Do Not Impute.

The results obtained by the classification methods depend on the previous imputation step, but also on the parameter configuration used by both the imputation and classification methods. Therefore they must be indicated in order to be able to reproduce any results obtained. In Subsection 3.4 the parameter configurations used by all the methods considered in this study are presented.

3.1 Data sets description

The experimentation has been carried out using 21 benchmark data sets from the UCI repository (Asuncion and Newman, 2007). Each data set is described by a set of characteristics such as the number of data samples, attributes and classes, summarized in Table 1. In this table, the percentage of MVs is indicated as well: the percentage of values which are missing, and the percentage of instances with at least one MV.

Data set	Acronym	# instances.	# attributes	# classes	% MV	% inst. with MV
Cleveland	CLE	303	14	5	0.14	1.98
Wisconsin	WIS	699	10	2	0.23	2.29
Credit	CRX	689	16	2	0.61	5.37
Breast	BRE	286	10	2	0.31	3.15
Autos	AUT	205	26	6	1.11	22.44
Primary tumor	PRT	339	18	21	3.69	61.06
Dermatology	DER	365	35	6	0.06	2.19
House-votes-84	HOV	434	17	2	5.3	46.54
Water-treatment	WAT	526	39	13	2.84	27.76
Sponge	SPO	76	46	12	0.63	28.95
Bands	BAN	540	40	2	4.63	48.7
Horse-colic	HOC	368	24	2	21.82	98.1
Audiology	AUD	226	71	24	1.98	98.23
Lung-cancer	LUN	32	57	3	0.27	15.63
Hepatitis	HEP	155	20	2	5.39	48.39
Mushroom	MUS	8124	23	2	1.33	30.53
Post-operative	POS	90	9	3	0.37	3.33
Echocardiogram	ECH	132	12	4	4.73	34.09
Soybean	SOY	307	36	19	6.44	13.36
Mammographic	MAM	961	6	2	2.81	13.63
Ozone	OZO	2534	73	2	8.07	27.11

Table 1: Data sets used

We cannot know anything about the randomness of MVs in the data sets, so we assume they are distributed in an *MAR* way, so the application of the imputation methods is feasible.

Most of the previous studies in Section 2.2 discard any previous natural missing data, if any, and then generate random MVs for their experiments with different percentages and MV distributions.

In our study we want to deal with the original MVs and therefore obtain the

real accuracy values of each data set with our imputation methods. Furthermore, the amount of data sets we have used is the largest of all missing data studies of Section 2.2. In addition to this, we use all kinds of data sets, which includes nominal data sets, numeric data sets and mixed-mode data sets.

In order to carry out the experimentation, we have used a 10-fold cross validation scheme. All the classification algorithms use the same partitions, to perform fair comparisons. We take the mean accuracy of training and test of the 10 partitions as a representative measure of the method's performance.

All these data sets have natural MVs, and we have imputed them with the following scheme. With the training partition, we apply the imputation method, extracting the relationships between the attributes, and filling in this partition. Next, with the information obtained, we fill in the MVs in the test partition. Since we have 14 imputation methods, we will obtain 14 instances of each partition of a given data set once they have been preprocessed. All these partitions will be used to train the classification methods used in our study, and then we will perform the test validation with the corresponding test partition. If the imputation method works only with numerical data, the nominal values are considered as a list of integer values, starting from 1 to the amount of different nominal values in the attribute.

3.2 Classification methods

In order to test the performance of the imputation methods, we have selected a set of representative classifiers. We can group them in three sub-categories. In Table 2 we summarize the classification methods we have used, organized in these three categories. The description of the former categories is as follows:

- The first group is the *Rule Induction Learning* category. This group refers to algorithms which infer rules using different strategies.
- The second group represents the *Approximate Models*. It includes Artificial Neural Networks, Support Vector Machines and Statistical Learning.
- The third and last group corresponds to the *Lazy Learning* category. This group incorporates methods which do not create any model, but use the training data to perform the classification directly.

Many of these classifiers appear in previous studies mentioned in Section 2.2. We have included an increased number of methods in our study (classical and currently most-used), so we can generalize better from the obtained results.

On the other hand, some methods do not work with numerical attributes (CN2, AQ and Naïve-Bayes). In order to discretize the numerical values, we have used the well-known discretizer proposed by Fayyad and Irani (1993).

For the SVM methods (C-SVM, ν-SVM and SMO), we have applied the usual preprocessing in the literature to these methods (Fan et al., 2005). This preprocessing consists of normalizing the numerical attributes to the $[0, 1]$ range, and binarizing the nominal attributes.

Method	Acronym	Refence
Rule Induction Learning		
C4.5	C4.5	(Quinlan, 1993)
Ripper	Ripper	(Cohen, 1995)
CN2	CN2	(Clark and Niblett, 1989)
AQ-15	AQ	(Michalksi, , Mozetic and Lavrac, 1986)
PART	PART	(Frank and Witten, 1998)
Slipper	Slipper	(Cohen and Singer, 1999)
Scalable Rule Induction Induction	SRI	(Pham and Afify, 2006)
Rule Induction Two In One	Ritio	(Wu and Urpani, 1999)
Rule Extraction System version 6	Rule-6	(Pham and Afify, 2005)
Approximate Models		
Multi-Layer Perceptron	MLP	(Moller, 1990)
C-SVM	C-SVM	(Fan, Chen and Lin, 2005)
ν-SVM	ν-SVM	(Fan et al., 2005)
Sequential Minimal Optimization	SMO	(Platt, 1999)
Radial Basis Function Network	RBFN	(Broomhead and Lowe, 1988)
RBFN Decremental	RBFND	(Broomhead and Lowe, 1988)
RBFN Incremental	RBFNI	(Plat, 1991)
Logistic	LOG	(le Cessie and van Houwelingen, 1992)
Naïve-Bayes	NB	(Domingos and Pazzani, 1997)
Learning Vector Quantization	LVQ	(Bezdek and Kuncheva, 2001)
Lazy Learning		
1-NN	1-NN	(McLachlan, 2004)
3-NN	3-NN	(McLachlan, 2004)
Locally Weighted Learning	LWL	(Atkeson, Moore and Schaal, 1997)
Lazy Learning of Bayesian Rules	LBR	(Zheng and Webb, 2000)

Table 2: Classifiers used by categories

3.3 Particular missing values treatment of the classification methods

Some of the presented classification methods in the previous section have their own MVs treatment:

- C4.5 uses a probabilistic approach to handling missing data. If there are missing values in an attribute X, C4.5 uses the subset with all known values of X to calculate the information gain. Once a test based on an attribute X is chosen, C4.5 uses a probabilistic approach to partition the instances with missing values in X. If the instance has an unknown value, this instance is assigned to all partitions with different weights for each one. The weight for the partition T_i is the probability that the instance belongs to T_i. This probability is estimated to be the sum of the weights of instances in T known to satisfy the test with outcome O_i, divided by the sum of weights of the cases in T with known values in the attribute X.

- CN2 algorithm uses a rather simple imputation method to treat missing data. Every missing value is filled in with its attribute's most common known value, before calculating the entropy measure.

Therefore, when using the DNI method, both C4.5 and CN2 will use their imputation abilities to treat the MVs. Therefore, we can compare their internal MVs treatment methods against the rest of the imputation methods from Section 2.1.

In the case of Neural Networks (MLP and RBFN variants), there are some interesting proposals for this case. Ennett, Frize and Walker (2001) proposed in their study to replace the MVs with "normal" values (i.e. replaced by zero) . This means that the MVs do not trigger the corresponding neuron which the MV is applied to, and the network can be trained on data with MVs, and evaluate instances with MVs as well.

The previously mentioned methods can handle the MVs in the case of the DNI method. On the other hand, the rest of the classification methods cannot handle the MVs. Thus we set the training and test accuracy to zero, as the method cannot build a model or compute a distance to the test instance.

3.4 Parameters used by the imputation and classification methods

In Table 3 we show the parameters used by each imputation method described in Section 2.1, in cases where the method needs a parameter. The values chosen are those recommended by their respective authors. Please refer to their respective papers for further descriptions of the parameters' meaning.

Method	Parameters
SVMI	Kernel= RBF C= 1.0 Epsilon= 0.001 shrinking= No
KNNI, WKNNI	K= 10
KMI	K= 10 iterations = 100 error = 100
FKMI	K= 3 iterations = 100 error = 100 m = 1.5
EC	T= 0.05
EM	iterations = 30 stagnation tolerance = 0.0001 inflation factor = 1 regression type = multiple ridge regression
SVDI	iterations = 30 stagnation tolerance = 0.005 inflation factor = 1 regression type = multiple ridge regression singular vectors = 10
LLSI	max number of nearest neighbor = 200

Table 3: Methods Parameters

In Table 4 the parameters used by the different classification methods are presented. All these parameters are the recommended ones that have been

Method	Parameters
C4.5	prune=true, confidence=0.25,instances per leaf=2
Ripper	grow percentage=0.66,K=2
CN2	percentage ex. To cover=0.95,star size=5,disjunt selectors=no
AQ	star size=5, disjunt selector=no
PART	confidence=0.25,intemsets per leaf=2
Slipper	grow percentage=0.66,K=2
SRI	beam width=8,min positives=2,min negatives=1
Ritio	-
Rule-6	beam width=5,min positives=2,min negatives=1
MLP	hidden layers=1, neurons per layer = 10
C-SVM	Kernel=poly.,C=100,eps=0.001,degree=1,gamma=0.01,coef0=0,p=1,shrink=yes
Nu-SVM	Kernel=poly.,nu=0.1,eps=0.001,degree=1,gamma=0.01,coef0=0,p=1,shrink=yes
SMO	C=1,tolerance=0.001,eps=1e-12,Kernel=polynomial,exp=1,lowerOrder=no
RBFN	neurons=50
RBFND	percent=0.1,initial neurons=20,alpha=0.3
RBFNI	epsilon=0.1,alpha=0.3,delta=0.5
LOG	ridge=1e-8,iteration limit=none
NB	-
LVQ	iterations=100,neurons=20,alpha=0.3,nu=0.8
1-NN	K=1,distance function=euclidean
3-NN	K=3,distance function=euclidean
LWL	K=3,Kernel function=constant
LBR	-

Table 4: Parameters used by the classification methods

extracted from the respective publications of the methods. Please refer to the associated publications listed in Table 2 to obtain the meaning of the different parameters.

4 Experimental results

In this section we analyze the experimental results obtained. We have created an associated webpage with all the results related to our analysis. The reason for this is to avoid long appendices, due to the size of the combination of all the imputation methods with the classification methods. The address is `http://sci2s.ugr.es/KAIS-MVDM/`. We have also included on this webpage the partitions of the used data sets for further comparisons. In order to compare the algorithms and MV methods we have used the Wilcoxon Signed Rank test, to support our analysis with a statistical test that provides us with statistical evidence of the good behavior of any approach. Therefore, the mentioned web page contains the following two documents:

- A document with all the accuracy results both in training and test for all the classification methods, each one with the 14 imputation methods.
- A document with a table summarizing the Wilcoxon test for all the imputation methods in respect to a determined classification method. The outcomes of the tables are based directly on the test accuracy results of the previous document.

The rest of the analysis is organized in two parts. First, we analyze all the methods together, without differentiating the groups. This approach is similar

to previous studies on the topic. Then, we have analyzed the methods organized by the different groups, obtaining different results. Therefore, the rest of this section is organized as follows:

- In Section 4.1 we introduce the comparison methodology used in the subsequent subsections.
- In Section 4.2 we show first the global results of the imputation methods for all the groups together.
- In Section 4.3 we study the behavior of the Rule Induction Learning classification methods.
- In Section 4.4 we analyze the Approximate methods.
- In Section 4.5 we compare the results of the imputation methods for the Lazy Learning algorithms.
- In Section 4.6 we summarize the suitability and performance of the imputation methods restricted to each group. In this way we intend to extract the best imputation method for each type of classifier, and analyze whether there is any kind of relation between them.

4.1 Comparison methodology

In order to appropriately analyze the imputation and classification methods, we use the Wilcoxon tables directly from the web page. These tables provide us with an average ranking for each imputation method. The content of the tables and its interpretation is as follows:

1. We create an $n \times n$ table for each classification method. In each cell, the outcome of the Wilcoxon signed rank test is shown.
2. In the aforementioned tables, if the p-value obtained by the Wilcoxon tests for a pair of imputation methods is higher than our α level, formerly 0.1, then we establish that there is a *tie* in the comparison (no significant difference was found), represented by a D.
3. If the p-value obtained by the Wilcoxon tests is lower than our α level, formerly 0.1, then we establish that there is a *win* (represented by a W) or a *loss* (represented by an L) in the comparison. If the method presented in the row has a better ranking than the method presented in the column in the Wilcoxon test then there is a *win*, otherwise there is a *loss*.

With these columns, we have produced an average ranking for each classifier. We have computed the number of times that an imputation methods wins, and the number of times that an imputation method wins and ties. Then we obtain the average ranking by putting those imputation methods which have a higher "wins + ties" sum first among the rest of the imputation methods. If a draw

is found for "wins + ties", we use the "wins" to establish the rank. If some methods obtain a draw for both "wins + ties" and "wins", then an average ranking is assigned for all of them.

In order to compare the imputation methods for the classification methods considered in each situation (global or family case), we have added two more final columns in the tables contained in the next subsections. In the first new column, we compute the mean of the rankings for each imputation method across all the classifiers of the correspondent group (column "Avg."), that is, the mean of every row. By doing so, we can obtain a new rank (final column RANKS), in which we propose a new ordering for the imputation methods for a given classifier's group, using the values of the column "Avg." to sort the imputation methods.

4.2 Results for all the classification methods

In this section, we analyze the different imputation approaches for all the imputation methods as a first attempt to obtain an "overall best" imputation method. Following the indications given in the previous subsection, in Table 5 the obtained average ranks and final imputation methods' rankings can be seen.

When comparing all the classifiers together, we find that it is difficult to establish differences between the imputation methods and to select the best one. The FKMI method obtains the best final ranking. However, the EC method has a very similar average ranking (5.70 for EC, 5.26 for FKMI). There are some additional methods that obtain a very similar average ranking, and they are not far from FKMI and EC. SVMI, KMI, MC and CMC have an average ranking between 6.09 and 6.28. Therefore we cannot firmly establish one best method from among all of them, and in this initial case we must consider a range of good possible imputation methods for the treatment of the MVs from the mentioned ones.

The DNI and IM methods do not obtain a good rank. In particular, the DNI method obtains a very high ranking (10.61) only exceeded by the BPCA imputation method which performs very badly. The IM method has an average rank, and it is situated in the middle of the ranking. Thus, we can consider discarding the examples with MVs, or not processing them, to be inadvisable, as expected from previous studies.

We must point out that the results obtained by the IM method should be considered with caution. Since several instances are discarded, the test and training partition tend to be smaller than the original ones. This allows the classifiers to obtain better results in training, since there are less instances and less noise from the MVs. In tests the classifier can achieve better results for some data sets if the remaining instances are well separated in the feature space, since a hit in the test partition counts for more in accuracy than in the other imputation methods (with complete test partitions).

From these results it is clear that we need to reduce the amount of classifiers when trying to obtain the best imputation method. In the following subsections

	RBFN	RBFND	RBFNI	C4.5	1-NN	LOG	LVQ	MLP	NB	ν-SVM	C-SVM	Ripper
IM	9	6.5	4.5	5	5	6	3.5	13	12	10	5.5	8.5
EC	1	1	1	2.5	9.5	3	7	8.5	10	13	1	8.5
KNNI	5	6.5	10.5	9	2.5	9	7	11	6.5	8	5.5	2.5
WKNNI	13	6.5	4.5	11	4	10	10	4.5	6.5	4.5	5.5	2.5
KMI	3.5	2	7	5	12	3	11	3	4.5	8	5.5	2.5
FKMI	12	6.5	10.5	7.5	6	3	1.5	4.5	11	4.5	5.5	2.5
SVMI	2	11.5	2.5	1	9.5	7.5	3.5	1.5	13	8	11	5.5
EM	3.5	6.5	13	13	11	12	12.5	10	4.5	4.5	10	12
SVDI	9	6.5	7	11	13	11	12.5	8.5	3	11.5	12	11
BPCA	14	14	14	14	14	13	7	14	2	2	13	13
LLSI	6	6.5	10.5	11	7.5	7.5	7	6.5	9	4.5	5.5	5.5
MC	9	6.5	10.5	7.5	7.5	3	7	6.5	8	11.5	5.5	8.5
CMC	9	13	2.5	5	1	3	1.5	1.5	14	14	5.5	8.5
DNI	9	11.5	7	2.5	2.5	14	14	12	1	1	14	14
PART	Slipper	3-NN	AQ	CN2	SMO	LBR	LWL	SRI	Ritio	Rule-6	Avg.	RANKS
1	4	11	6.5	10	5.5	5	8	6.5	6	5	6.83	7
6.5	1	13	6.5	5.5	2	9	8	6.5	6	1	5.70	2
6.5	11	5.5	11	5.5	5.5	9	8	11.5	11	11	7.76	10
6.5	7	5.5	6.5	1	5.5	9	8	11.5	6	11	6.96	8
6.5	3	5.5	6.5	5.5	9	9	2.5	9.5	12	7.5	6.24	5
6.5	10	1.5	2	5.5	3	9	2.5	1	2	3	5.26	1
6.5	7	9	1	5.5	9	3	8	6.5	6	2	6.09	3
6.5	7	5.5	12	13	11.5	9	2.5	3	6	4	8.37	11
6.5	12	12	10	12	11.5	1	12	9.5	10	11	9.72	12
13	7	14	13	14	13	13	13	13	13	13	11.87	14
6.5	7	5.5	6.5	11	9	9	8	3	6	7.5	7.22	9
6.5	2	1.5	6.5	5.5	5.5	3	2.5	3	6	7.5	6.11	4
12	13	5.5	3	5.5	1	3	8	6.5	1	7.5	6.28	6
14	14	10	14	5.5	14	14	14	14	14	14	10.61	13

Table 5: Average ranks for all the classifiers

we have focused on the different types of classification methods in order to avoid the high ranking variation observed in Table 5.

4.3 Results for the Rule Induction Learning methods

In this section, we present the results of the Rule Induction classification methods. In Table 6 we show the ranking for each classification method belonging to this group. This table's structure is the same as that described in Subsection 4.1. Therefore, we only perform the average between the rankings obtained for the classification algorithms belonging to this group.

	C45	Ripper	PART	Slipper	AQ	CN2	SRI	Ritio	Rules-6	Avg.	RANKS
IM	5	8.5	1	4	6.5	10	6.5	6	5	5.83	4
EC	2.5	8.5	6.5	1	6.5	5.5	6.5	6	1	4.89	3
KNNI	9	2.5	6.5	11	11	5.5	11.5	11	11	8.78	11
WKNNI	11	2.5	6.5	7	6.5	1	11.5	6	11	7.00	8
KMI	5	2.5	6.5	3	6.5	5.5	9.5	12	7.5	6.44	6
FKMI	7.5	2.5	6.5	10	2	5.5	1	2	3	4.44	1
SVMI	1	5.5	6.5	7	1	5.5	6.5	6	2	4.56	2
EM	13	12	6.5	7	12	13	3	6	4	8.50	10
SVDI	11	11	6.5	12	10	12	9.5	10	11	10.33	12
BPCA	14	13	13	7	13	14	13	13	13	12.56	14
LLSI	11	5.5	6.5	7	6.5	11	3	6	7.5	7.11	9
MC	7.5	8.5	6.5	2	6.5	5.5	3	6	7.5	5.89	5
CMC	5	8.5	12	13	3	5.5	6.5	1	7.5	6.89	7
DNI	2.5	14	14	14	14	5.5	14	14	14	11.78	13

Table 6: Average ranks for the Rule Induction Learning methods

We can observe that, for the Rule Induction Learning classifiers, the imputation methods FKMI, SVMI and EC perform best. The differences between these three methods in average rankings are low. Thus we can consider that these three imputation methods are the most suitable for this kind of classifier. They are well separated from the other imputation methods and we cannot choose a best method from among these three. This is in contrast to the global results presented in Subsection 4.2, where little differences could be found among the first ranked imputation methods. Both FKMI and EC methods were also considered among the best in the global first approach presented in the previous subsection.

On the other hand, BPCA and DNI are the worst methods. The BPCA method usually performs badly for all the classifiers. As DNI is also a bad option, this means that the Rule Induction Learning algorithms would greatly benefit from the use of the imputation methods, despite some of them being capable of dealing with MVs on their own.

The rest of the imputation methods span between an average rank of 5.8 to 9, with a great difference between the BPCA and DNI methods in ranking. The IM method is fourth, and that could mean that the Rule Induction Learning algorithms perform better with complete instances in training and test, but they do not work well with test instances with imputed MVs. However, avoiding test cases with MVs is not always possible.

4.4 Results for the Approximate methods

In this section we present the obtained results for the Approximate models. In Table 7 we can observe the rankings associated with the methods belonging to this group. Again, this table structure is the same as described in Subsection 4.1.

	RBFN	RBFND	RBFNI	LOG	LVQ	MLP	NB	ν-SVM	C-SVM	SMO	Avg.	RANKS
IM	9	6.5	4.5	6	3.5	13	12	10	5.5	5.5	7.55	10
EC	1	1	1	3	7	8.5	10	13	1	2	4.75	1
KNNI	5	6.5	10.5	9	7	11	6.5	8	5.5	5.5	7.45	9
WKNNI	13	6.5	4.5	10	10	4.5	6.5	4.5	5.5	5.5	7.05	6
KMI	3.5	2	7	3	11	3	4.5	8	5.5	9	5.65	2
FKMI	12	6.5	10.5	3	1.5	4.5	11	4.5	5.5	3	6.20	3
SVMI	2	11.5	2.5	7.5	3.5	1.5	13	8	11	9	6.95	5
EM	3.5	6.5	13	12	12.5	10	4.5	4.5	10	11.5	8.80	11
SVDI	9	6.5	7	11	12.5	8.5	3	11.5	12	11.5	9.25	12
BPCA	14	14	14	13	7	14	2	2	13	13	10.60	14
LLSI	6	6.5	10.5	7.5	7	6.5	9	4.5	5.5	9	7.20	7
MC	9	6.5	10.5	3	7	6.5	8	11.5	5.5	5.5	7.30	8
CMC	9	13	2.5	3	1.5	1.5	14	14	5.5	1	6.50	4
DNI	9	11.5	7	14	14	12	1	1	14	14	9.75	13

Table 7: Average ranks for the Approximate methods

In the case of the Approximate models, the differences between imputation methods are even more evident. We can select the EC method as the best solution, as it has a difference of ranking of almost 1 with KMI, which stands as the second best. This difference increases when considering the third best, FKMI. No other family of classifiers present this gap in the rankings. Therefore, in this family of classification methods we could, with some confidence, establish the EC method as the best choice. This is in contrast with the global results, from which there is no outstanding method.

The DNI and IM methods are among the worst. This means that for the Approximate methods the use of some kind of MV treatment is mandatory, whereas the EC method is the most suitable one. As with the Rule Induction Learning methods, the BPCA method is the worst choice, with the highest ranking.

4.5 Results for the Lazy Learning methods

The results for the last group are presented in Table 8. Again, this table structure is the same as described in Subsection 4.1.

For the Lazy Learning models, the MC method is the best with the lowest average ranking. The CMC method, which is relatively similar to MC, also obtains a low rank very close to MC's. Only the FKMI method obtains a low enough rank to be compared with the MC and CMC methods. The rest of the imputation methods are far from these lowest ranks with almost two points of difference in the ranking. This situation is similar to the Rule Induction Learning methods' family, in which we could find three outstanding methods with a difference of 1 between the 3rd and 4th ones.

	1-NN	3-NN	LBR	LWL	Avg.	RANKS
IM	5	11	5	8	7.25	7
EC	9.5	13	9	8	9.88	12
KNNI	2.5	5.5	9	8	6.25	4
WKNNI	4	5.5	9	8	6.63	5
KMI	12	5.5	9	2.5	7.25	8
FKMI	6	1.5	9	2.5	4.75	3
SVMI	9.5	9	3	8	7.38	9
EM	11	5.5	9	2.5	7.00	6
SVDI	13	12	1	12	9.50	11
BPCA	14	14	13	13	13.50	14
LLSI	7.5	5.5	9	8	7.50	10
MC	7.5	1.5	3	2.5	3.63	1
CMC	1	5.5	3	8	4.38	2
DNI	2.5	10	14	14	10.13	13

Table 8: Average ranks for the Lazy Learning methods

Again, the DNI and IM methods obtain high rankings. The DNI method is one of the worst, with only the BPCA method performing worse. As with the Approximate models, the imputation methods produce a significant improvement in the accuracy of these classification methods and they should always be considered prior to their application.

4.6 Summary of the group-based results

In the previous Subsections 4.3, 4.4 and 4.5 we have observed that when comparing the imputation methods to similar classifiers, more significant information about the best ones can be extracted. In this section we summarize these best imputation methods for each group, and we analyze the similarity between them. For the Wilcoxon tables with their rankings from Subsections 4.3 to 4.5, we have built Table 9 with the best three methods of each group. We have stressed in bold those rankings equal to or below three.

	Rule I. Learning RANKING	**Approx. models** RANKING	**Lazy L. models** RANKING
EC	**3**	**1**	12
KMI	6	**2**	8
FKMI	**1**	**3**	**3**
SVMI	**2**	5	9
MC	5	8	**1**
CMC	7	4	**2**

Table 9: Best imputation methods for each group

From Table 9 we can observe some interesting aspects:

- The Rule Induction Learning category and the Approximate models share the EC and FKMI methods in their top 3 best imputation algorithms.
- The Lazy Learning models only share the FKMI method in common with

the rest. This means that the best method obtained in the general analysis in Section 4.2 is the only one present in all of the three groups as one of the best.

- The CMC and MC methods do not perform outstandingly in the Rule Induction Learning methods and Approximate models. Thus, we can consider that such a simple imputation strategy is only useful when we do not have to build any model from the data.

From these results, we can obtain a set of imputation methods for each group that represent the best option(s) for them. Notice that the results for all of the three groups do not correspond to the global result from the previous section. However, the FKMI imputation method is a good option in all three groups, even if it is not the winner. The EC method is also a good option except for the Lazy Learning methods. Therefore, we can establish that the consideration of different imputation methods is required in each case

It is important to notice that DNI and IM methods are never the best or among the best imputation methods for any group. Only in the case of the Rule Induction Learning methods does the IM imputation method obtain a relatively low rank (4th place) as we have previously mentioned. This fact indicates that the imputation methods usually outperform the non-imputation strategies.

As a final remark, we can state that the results obtained in our study cohere with those mentioned in Subsection 2.2, and particularly with Acuna and Rodriguez (2004), Batista and Monard (2003), Farhangfar et al. (2008), Feng et al. (2005), García-Laencina et al. (2009), Twala (2009) and Li et al. (2004); that is:

- The imputation methods which fill in the MVs outperform the case deletion (IM method) and the lack of imputation (DNI method).

- There is no universal imputation method which performs best for all classifiers.

Please note that we have tackled the second point by adding a categorization and a wide benchmark bed, obtaining a group of recommended imputation methods for each family.

5 Influence of the imputation on the instances and individual features

In the previous section we have analyzed the relationship between the use of several imputation methods with respect to the classifiers' accuracy. However, it would be interesting to relate the influence of the imputation methods to the information contained in the data set. In order to study the influence and the benefits/drawbacks of using the different imputation methods, we have considered the use of two different measures. They are described as follows:

- Wilson's Noise Ratio: This measure proposed by Wilson (1972) observes the noise in the data set. For each instance of interest, the method looks for the K nearest neighbors (using the euclidean distance), and uses the class labels of such neighbors in order to classify the considered instance. If the instance is not correctly classified, then the variable *noise* is increased by one unit. Therefore, the final noise ratio will be

$$\text{Wilson's Noise} = \frac{noise}{\#\text{ instances in the data set}}$$

  In particular, we only compute the noise for the *imputed* instances considering $K = 5$.

- Mutual Information: Mutual information (MI) is considered to be a good indicator of relevance between two random variables (Cover and Thomas, 1991). Recently, the use of the MI measure in feature selection has become well-known and seen to be successful (Kwak and Choi, 2002*b*; Kwak and Choi, 2002*a*; Peng, Long and Ding, 2005). The use of the MI measure for continuous attributes has been tackled by (Kwak and Choi, 2002*a*), allowing us to compute the MI measure not only in nominal-valued data sets.
  In our approach, we calculate the MI between each input attribute and the class attribute, obtaining a set of values, one for each input attribute. In the next step we compute the ratio between each one of these values, considering the imputation of the data set with one imputation method in respect to the not imputed data set. The average of these ratios will show us if the imputation of the data set produces a gain in information:

$$\text{Avg. MI Ratio} = \frac{\sum_{x_i \in X} \frac{MI_\alpha(x_i)+1}{MI(x_i)+1}}{|X|}$$

  where X is the set of input attributes, $MI_\alpha(i)$ represents the MI value of the ith attribute in the imputed data set and $MI(i)$ is the MI value of the ith input attribute in the not imputed data set. We have also applied the Laplace correction, summing 1 to both numerator and denominator, as an MI value of zero is possible for some input attributes.
  The calculation of $MI(x_i)$ depends on the type of attribute x_i. If the attribute x_i is nominal, the MI between x_i and the class label Y is computed as follows:

$$MI_{nominal}(x_i) = I(x_i; Y) = \sum_{z \in x_i} \sum_{y \in Y} p(z, y) log_2 \frac{p(z, y)}{p(z)p(y)}.$$

  On the other hand, if the attribute x_i is numeric, we have used the Parzen window density estimate as shown in (Kwak and Choi, 2002*a*) considering a Gaussian window function:

$$MI_{numeric}(x_i) = I(x_i; Y) = H(Y) - H(C|X);$$

where $H(Y)$ is the entropy of the class label

$$H(Y) = -\sum_{y \in Y} p(y) log_2 p(y);$$

and $H(C|X)$ is the conditional entropy

$$H(Y|x_i) = -\sum_{z \in x_i} \sum_{y \in Y} p(z,y) log_2 p(y|z).$$

Considering that each sample has the same probability, applying the Bayesian rule and approximating $p(y|z)$ by the Parzen window we get:

$$\hat{H}(Y|x_i) = -\sum_{j=1}^{n} \frac{1}{n} \sum_{y=1}^{N} \hat{p}(y|z_j) log_2 \hat{p}(y|z_j)$$

where n is the number of instances in the data set, N is the total number of class labels and $\hat{p}(c|x)$ is

$$\hat{p}(y|z) = \frac{\sum_{i \in I_c} exp\left(-\frac{(z-z_i)\Sigma^{-1}(z-z_i)}{2h^2}\right)}{\sum_{k=1}^{N} \sum_{i \in I_k} exp\left(-\frac{(z-z_i)\Sigma^{-1}(z-z_i)}{2h^2}\right)}.$$

In this case, I_c is the set of indices of the training examples belonging to class c, and Σ is the covariance of the random variable $(z - z_i)$.

Comparing with Wilson's noise ratio we can observe which imputation methods reduce the impact of the MVs as a noise, and which methods produce noise when imputing. In addition the MI ratio allows us to relate the attributes to the imputation results. A value of the MI ratio higher than 1 will indicate that the imputation is capable of relating more of the attributes individually to the class labels. A value lower than 1 will indicate that the imputation method is adversely affecting the relationship between the individual attributes and the class label.

In Table 10 we have summarized the Wilson's noise ratio values for the 21 data sets considered in our study. We must point out that the results of Wilson's noise ratio are related to a given data set. Hence, the characteristics of the proper data appear to determine the values of this measure.

In Table 11 we have summarized the average MI ratios for the 21 data sets. In the results we can observe that the average ratios are usually close to 1; that is, the use of imputation methods appears to harm the relationship between the class label and the input attribute little or not at all, even improving it in some cases. However, the mutual information considers only one attribute at a time and therefore the relationships between the input attributes are ignored. The imputation methods estimate the MVs using such relationships and can afford improvements in the performance of the classifiers. Hence the highest values of average MI ratios could be related to those methods which can obtain better

Table 10: Wilson's noise ratio values

Data-set	Imp. Method	% Wilson's Noise Ratio	Data-set	Imp. Method	% Wilson's Noise Ratio	Data-set	Imp. Method	% Wilson's Noise Ratio
CLE	MC	50.0000	HOV	MC	7.9208	HEP	MC	17.3333
	CMC	50.0000		CMC	5.4455		CMC	**16.0000**
	KNNI	50.0000		KNNI	7.4257		KNNI	20.0000
	WKNNI	50.0000		WKNNI	7.4257		WKNNI	20.0000
	KMI	50.0000		KMI	7.4257		KMI	20.0000
	FKMI	50.0000		FKMI	7.9208		FKMI	17.3333
	SVMI	50.0000		SVMI	6.9307		SVMI	17.3333
	EM	66.6667		EM	11.8812		EM	22.6667
	SVDI	66.6667		SVDI	8.9109		SVDI	21.3333
	BPCA	50.0000		BPCA	6.9307		BPCA	21.3333
	LLSI	50.0000		LLSI	**4.9505**		LLSI	18.6667
	EC	**33.3333**		EC	7.4257		EC	**16.0000**
WIS	MC	18.7500	WAT	MC	31.5068	MUS	MC	**0.0000**
	CMC	**12.5000**		CMC	**21.2329**		CMC	**0.0000**
	KNNI	**12.5000**		KNNI	27.3973		KNNI	**0.0000**
	WKNNI	**12.5000**		WKNNI	27.3973		WKNNI	**0.0000**
	KMI	**12.5000**		KMI	27.3973		KMI	**0.0000**
	FKMI	**12.5000**		FKMI	31.5068		FKMI	**0.0000**
	SVMI	**12.5000**		SVMI	23.9726		SVMI	**0.0000**
	EM	**12.5000**		EM	46.5753		EM	**0.0000**
	SVDI	**12.5000**		SVDI	49.3151		SVDI	**0.0000**
	BPCA	**12.5000**		BPCA	26.0274		BPCA	**0.0000**
	LLSI	**12.5000**		LLSI	25.3425		LLSI	**0.0000**
	EC	**12.5000**		EC	22.6027		EC	**0.0000**
CRX	MC	18.9189	SPO	MC	27.2727	POS	MC	**33.3333**
	CMC	18.9189		CMC	**22.7273**		CMC	**33.3333**
	KNNI	21.6216		KNNI	27.2727		KNNI	**33.3333**
	WKNNI	21.6216		WKNNI	27.2727		WKNNI	**33.3333**
	KMI	21.6216		KMI	27.2727		KMI	**33.3333**
	FKMI	18.9189		FKMI	27.2727		FKMI	**33.3333**
	SVMI	13.5135		SVMI	27.2727		SVMI	**33.3333**
	EM	32.4324		EM	36.3636		EM	**33.3333**
	SVDI	27.0270		SVDI	31.8182		SVDI	**33.3333**
	BPCA	21.6216		BPCA	27.2727		BPCA	**33.3333**
	LLSI	18.9189		LLSI	27.2727		LLSI	**33.3333**
	EC	**13.5135**		EC	27.2727		EC	**33.3333**
BRE	MC	55.5556	BAN	MC	25.4753	ECH	MC	40.0000
	CMC	55.5556		CMC	24.3346		CMC	40.0000
	KNNI	55.5556		KNNI	23.1939		KNNI	46.6667
	WKNNI	55.5556		WKNNI	22.8137		WKNNI	44.4444
	KMI	55.5556		KMI	25.4753		KMI	46.6667
	FKMI	55.5556		FKMI	24.3346		FKMI	40.0000
	SVMI	55.5556		SVMI	**21.2928**		SVMI	44.4444
	EM	**44.4444**		EM	26.2357		EM	51.1111
	SVDI	**44.4444**		SVDI	22.4335		SVDI	48.8889
	BPCA	66.6667		BPCA	23.9544		BPCA	44.4444
	LLSI	66.6667		LLSI	24.7148		LLSI	**37.7778**
	EC	66.6667		EC	23.5741		EC	48.8889
AUT	MC	45.6522	HOC	MC	19.3906	SOY	MC	**2.4390**
	CMC	41.3043		CMC	**10.2493**		CMC	**2.4390**
	KNNI	41.3043		KNNI	20.2216		KNNI	**2.4390**
	WKNNI	41.3043		WKNNI	19.1136		WKNNI	**2.4390**
	KMI	41.3043		KMI	21.8837		KMI	**2.4390**
	FKMI	45.6522		FKMI	20.4986		FKMI	**2.4390**
	SVMI	43.4783		SVMI	20.2216		SVMI	**2.4390**
	EM	58.6957		EM	21.0526		EM	**2.4390**
	SVDI	52.1739		SVDI	21.0526		SVDI	7.3171
	BPCA	43.4783		BPCA	19.3906		BPCA	7.3171
	LLSI	45.6522		LLSI	20.4986		LLSI	**2.4390**
	EC	**30.4348**		EC	20.7756		EC	**2.4390**
PRT	MC	71.0145	AUD	MC	38.7387	MAM	MC	21.3740
	CMC	**60.8696**		CMC	**32.8829**		CMC	**13.7405**
	KNNI	69.5652		KNNI	38.7387		KNNI	25.9542
	WKNNI	69.5652		WKNNI	38.7387		WKNNI	25.9542
	KMI	71.0145		KMI	38.7387		KMI	24.4275
	FKMI	71.0145		FKMI	38.7387		FKMI	20.6107
	SVMI	68.1159		SVMI	37.8378		SVMI	16.7939
	EM	88.4058		EM	53.6036		EM	20.6107
	SVDI	91.7874		SVDI	46.3964		SVDI	27.4809
	BPCA	71.4976		BPCA	40.5405		BPCA	25.1908
	LLSI	69.5652		LLSI	36.9369		LLSI	26.7176
	EC	66.1836		EC	37.8378		EC	18.3206
DER	MC	**0.0000**	LUN	MC	80.0000	OZO	MC	4.8035
	CMC	**0.0000**		CMC	80.0000		CMC	**3.6390**
	KNNI	**0.0000**		KNNI	80.0000		KNNI	4.3668
	WKNNI	**0.0000**		WKNNI	80.0000		WKNNI	4.5124
	KMI	**0.0000**		KMI	80.0000		KMI	4.9491
	FKMI	**0.0000**		FKMI	80.0000		FKMI	4.0757
	SVMI	**0.0000**		SVMI	80.0000		SVMI	3.7846
	EM	**0.0000**		EM	**20.0000**		EM	4.8035
	SVDI	**0.0000**		SVDI	40.0000		SVDI	4.8035
	BPCA	**0.0000**		BPCA	80.0000		BPCA	4.3668
	LLSI	**0.0000**		LLSI	80.0000		LLSI	4.2213
	EC	**0.0000**		EC	80.0000		EC	4.8035

estimates for the MVs, and maintaining the relationship degree between the class labels and the isolated input attributes. It is interesting to note that when analyzing the MI ratio, the values do not appear to be as highly data dependant as Wilson's noise ratio, as the values for all the data sets are more or less close to each other.

If we count the methods with the lowest Wilson's noise ratios in each data set in Table 10, we find that the CMC method is first, with 12 times the lowest one, and the EC method is second with 9 times the lowest one. If we count the methods with the highest mutual information ratio in each data set in Table 11, the EC method has the highest ratio for 7 data sets and is therefore the first one. The CMC method has the highest ratio for 5 data sets and is the second one in this case. Considering the analysis of the previous Subsection 4.6 with these two methods:

- The EC method is the best method obtained for the Approximative models, and the third best for the Rule Induction Learning methods. In the latter case, the average ranking of EC is 4.89, very close to the average ranking 4.44 and 4.56 of FKMI and SVMI respectively.
- The CMC method is the second best method for the Lazy Learning models, and very close to the first one (MC) with an average ranking of 3.63.

Next, we rank all the imputation methods according to the values presented in Tables 10 and 11. In order to do so, we have calculated the average rankings of each imputation method for all the data sets, for both Wilson's noise ratio and the mutual information ratio. The method to compute this average ranking is the same as that presented in Subsection 4.2. In Table 12 we have gathered together these average rankings, as well as their relative position in parentheses.

From the average rankings shown in Table 12, we can observe that the CMC method is the first for both rankings. The EC method is the second for the mutual information ratio, and the third one for Wilson's noise ratio. The SVMI method obtains the second lowest ranking for Wilson's noise ratio, and the fourth lowest ranking for the MI ratio. The SVMI method is the second best method for the Rule Induction Learning algorithms with average rankings close to EC.

With the analysis performed we have quantified the noise induced by the imputation methods and how the relationship between each input attribute and the class is maintained. We have discovered that the CMC and EC methods show good behavior for these two measures, and they are two methods that provide good results for an important range of learning methods, as we have previously analyzed. In short, these two approaches introduce less noise and maintain the mutual information better. They can provide us with a first characterization of imputation methods and a first step for providing us with tools for analyzing the imputation method's behavior.

Table 11: Average mutual information ratio

Data-set	Imp. Method	Avg. MI ratio	Data-set	Imp. Method	Avg. MI ratio	Data-set	Imp. Method	Avg. MI ratio
CLE	MC	0.998195	HOV	MC	0.961834	HEP	MC	0.963765
	CMC	0.998585		CMC	1.105778		CMC	0.990694
	KNNI	0.998755		KNNI	0.965069		KNNI	0.978564
	WKNNI	0.998795		WKNNI	0.965069		WKNNI	0.978343
	KMI	0.998798		KMI	0.961525		KMI	0.980094
	FKMI	0.998889		FKMI	0.961834		FKMI	0.963476
	SVMI	0.998365		SVMI	0.908067		SVMI	1.006819
	EM	0.998152		EM	0.891668		EM	0.974433
	SVDI	0.997152		SVDI	0.850361		SVDI	0.967673
	BPCA	0.998701		BPCA	1.091675		BPCA	0.994420
	LLSI	0.998882		LLSI	**1.122904**		LLSI	0.995464
	EC	**1.000148**		EC	1.007843		EC	**1.024019**
WIS	MC	0.999004	WAT	MC	0.959488	MUS	MC	1.018382
	CMC	0.999861		CMC	0.967967		CMC	1.018382
	KNNI	0.999205		KNNI	0.961601		KNNI	0.981261
	WKNNI	0.999205		WKNNI	0.961574		WKNNI	0.981261
	KMI	0.999322		KMI	0.961361		KMI	1.018382
	FKMI	0.998923		FKMI	0.961590		FKMI	1.018382
	SVMI	0.999412		SVMI	0.967356		SVMI	0.981261
	EM	0.990030		EM	0.933846		EM	**1.142177**
	SVDI	0.987066		SVDI	0.933040		SVDI	1.137152
	BPCA	0.998951		BPCA	0.964255		BPCA	0.987472
	LLSI	0.999580		LLSI	0.964063		LLSI	0.977275
	EC	**1.000030**		EC	**1.027369**		EC	1.017366
CRX	MC	1.000883	SPO	MC	0.997675	POS	MC	1.012293
	CMC	1.000966		CMC	**1.022247**		CMC	1.012293
	KNNI	0.998823		KNNI	0.999041		KNNI	1.012293
	WKNNI	0.998870		WKNNI	0.999041		WKNNI	1.012293
	KMI	1.001760		KMI	0.998464		KMI	1.012293
	FKMI	1.000637		FKMI	0.997675		FKMI	1.012293
	SVMI	0.981878		SVMI	1.015835		SVMI	1.012293
	EM	0.985609		EM	0.982325		EM	1.012293
	SVDI	0.976398		SVDI	0.979187		SVDI	1.014698
	BPCA	0.999934		BPCA	1.006236		BPCA	1.012293
	LLSI	1.001594		LLSI	1.004821		LLSI	**1.018007**
	EC	**1.008718**		EC	1.018620		EC	0.997034
BRE	MC	0.998709	BAN	MC	1.012922	ECH	MC	0.981673
	CMC	0.998709		CMC	1.070857		CMC	0.995886
	KNNI	0.992184		KNNI	0.940369		KNNI	0.997912
	WKNNI	0.992184		WKNNI	0.940469		WKNNI	**0.998134**
	KMI	0.998709		KMI	1.016101		KMI	0.967169
	FKMI	0.998709		FKMI	1.020989		FKMI	0.983606
	SVMI	0.998709		SVMI	**1.542536**		SVMI	0.987678
	EM	**1.013758**		EM	1.350315		EM	0.967861
	SVDI	0.999089		SVDI	1.365572		SVDI	0.935855
	BPCA	1.000201		BPCA	1.010596		BPCA	0.972327
	LLSI	1.000201		LLSI	1.015033		LLSI	0.988591
	EC	1.001143		EC	1.102328		EC	0.970029
AUT	MC	0.985610	HOC	MC	0.848649	SOY	MC	1.056652
	CMC	0.991113		CMC	**2.039992**		CMC	1.123636
	KNNI	0.986239		KNNI	0.834734		KNNI	1.115818
	WKNNI	0.985953		WKNNI	0.833982		WKNNI	1.115818
	KMI	0.985602		KMI	0.821936		KMI	1.056652
	FKMI	0.984694		FKMI	0.849141		FKMI	1.056652
	SVMI	0.991850		SVMI	0.843456		SVMI	**1.772589**
	EM	0.970557		EM	0.775773		EM	1.099286
	SVDI	0.968938		SVDI	0.750930		SVDI	1.065865
	BPCA	0.986631		BPCA	0.964587		BPCA	1.121603
	LLSI	0.985362		LLSI	0.926068		LLSI	1.159610
	EC	**1.007652**		EC	0.911543		EC	1.222631
PRT	MC	0.949896	AUD	MC	0.990711	MAM	MC	0.974436
	CMC	**1.120006**		CMC	1.032162		CMC	1.029154
	KNNI	0.976351		KNNI	0.993246		KNNI	0.965926
	WKNNI	0.976351		WKNNI	0.993246		WKNNI	0.965926
	KMI	0.949896		KMI	1.000235		KMI	0.966885
	FKMI	0.949896		FKMI	0.990711		FKMI	0.974228
	SVMI	1.038152		SVMI	1.007958		SVMI	**1.272993**
	EM	0.461600		EM	1.129168		EM	0.980865
	SVDI	0.485682		SVDI	1.065091		SVDI	1.052790
	BPCA	0.987598		BPCA	1.156676		BPCA	0.978209
	LLSI	1.016230		LLSI	1.061197		LLSI	0.994349
	EC	1.053185		EC	**1.209608**		EC	1.269505
DER	MC	1.000581	LUN	MC	0.996176	OZO	MC	0.982873
	CMC	**1.002406**		CMC	1.008333		CMC	**0.989156**
	KNNI	0.999734		KNNI	0.996176		KNNI	0.982759
	WKNNI	0.999734		WKNNI	0.996176		WKNNI	0.982721
	KMI	1.000581		KMI	0.996176		KMI	0.982495
	FKMI	1.000581		FKMI	0.996176		FKMI	0.982951
	SVMI	1.001566		SVMI	1.006028		SVMI	0.988297
	EM	1.000016		EM	1.067844		EM	0.979977
	SVDI	0.999691		SVDI	**1.076334**		SVDI	0.979958
	BPCA	0.999633		BPCA	0.996447		BPCA	0.983318
	LLSI	0.999170		LLSI	1.007612		LLSI	0.983508
	EC	1.000539		EC	1.002385		EC	0.944747

Table 12: Average rankings for Wilson's noise ratio and Mutual information ratio

	Avg. Rankings	
	Wilson's noise ratio	Mutual information
MC	6.98 (8)	8.05 (11)
CMC	**3.79** (1)	**3.60** (1)
KNNI	6.43 (7)	7.69 (8)
WKNNI	6.17 (5)	7.79 (9)
KMI	7.38 (10)	7.60 (6)
FKMI	6.36 (6)	7.62 (7)
SVMI	4.67 (2)	4.90 (4)
EM	8.93 (12)	7.90 (10)
SVDI	8.86 (11)	8.48 (12)
BPCA	7.17 (9)	5.79 (5)
LLSI	5.98 (4)	4.74 (3)
EC	**5.31** (3)	**3.86** (2)

6 Lessons learned

This study is a general comparison of classification methods not previously considered in MV studies, arranged into three different groups. The results obtained agree with previous studies:

- The imputation methods which fill in the MVs outperform the case deletion (IM method) and the lack of imputation (DNI method).
- There is no universal imputation method which performs best for all classifiers.

As a global imputation method, from the results seen in Section 4.2, the use of the *FKMI* and *EC* imputation methods are the best choices.

From the obtained results in Section 4.6, the particular analysis of the MVs treatment methods conditioned to the classification methods' groups is necessary. Thus, we can stress particular imputation algorithms based on the classification groups, as in the case of the *FKMI* method for the Rule Induction Learning group, the *EC* method for the Approximate Models and the *MC* method for the Lazy Learning model. Therefore, we can confirm the positive effect of the imputation methods and the classifiers' behavior, and the presence of more suitable imputation methods for some particular classifier categories than others.

Moreover, in Section 5 we have analyzed the influence of the imputation methods in respect to two measures. These two measures are the *Wilson's noise ratio* and the *average mutual information difference*. The first one quantifies the noise induced by the imputation method in the instances which contain MVs. The second one examines the increment or decrement in the relationship of the

isolated input attributes with respect to the class label. We have observed that the CMC and EC methods are the ones which introduce less noise and maintain the mutual information better.

Acknowledgements

This work was supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01. J. Luengo holds a FPU scholarship from Spanish Ministry of Education and Science.

References

Acuna, E. and Rodriguez, C. (2004), *Classification, Clustering and Data Mining Applications*, Springer-Verlag Berlin-Heidelberg, pp. 639–648.

Alcalá-fdez, J., Sánchez, L., García, S., Jesus, M. J. D., Ventura, S., Garrell, J. M., Otero, J., Bacardit, J., Rivas, V. M., Fernández, J. C. and Herrera, F. (2009), 'Keel: A software tool to assess evolutionary algorithms for data mining problems', *Soft Computing* **13**(3), 307–318.

Asuncion, A. and Newman, D. (2007), 'UCI machine learning repository'. **URL:** *http://archive.ics.uci.edu/ml/*

Atkeson, C. G., Moore, A. W. and Schaal, S. (1997), 'Locally weighted learning', *Artificial Intelligence Review* **11**, 11–73.

Barnard, J. and Meng, X. (1999), 'Applications of multiple imputation in medical studies: From aids to nhanes', *Stat. Methods Med. Res.* **8**(1), 17–36.

Batista, G. and Monard, M. (2003), 'An analysis of four missing data treatment methods for supervised learning', *Applied Artificial Intelligence* **17**(5), 519–533.

Bezdek, J. and Kuncheva, L. (2001), 'Nearest prototype classifier designs: An experimental study', *International Journal of Intelligent Systems* **16**(12), 1445–1473.

Broomhead, D. and Lowe, D. (1988), 'Multivariable functional interpolation and adaptive networks', *Complex Systems* **11**, 321–355.

Clark, P. and Niblett, T. (1989), 'The cn2 induction algorithm', *Machine Learning Journal* **3**(4), 261–283.

Cohen, W. and Singer, Y. (1999), A simple and fast and and effective rule learner, *in* 'Proceedings of the Sixteenth National Conference on Artificial Intelligence', pp. 335–342.

Cohen, W. (1995), Fast effective rule induction, *in* 'Machine Learning: Proceedings of the Twelfth International Conference', pp. 1–10.

Cover, T. M. and Thomas, J. A. (1991), *Elements of Information Theory*, 2 edn, John Wiley.

Demšar, J. (2006), 'Statistical comparisons of classifiers over multiple data sets', *Journal of Machine Learning Research* **7**, 1–30.

Ding, Y. and Simonoff, J. S. (2010), 'An investigation of missing data methods for classification trees applied to binary response data', *Journal of Machine Learning Research* **11**, 131–170.

Domingos, P. and Pazzani, M. (1997), 'On the optimality of the simple bayesian classifier under zero-one loss', *Machine Learning* **29**, 103–137.

Ennett, C. M., Frize, M. and Walker, C. R. (2001), 'Influence of missing values on artificial neural network performance', *Stud Health Technol Inform* **84**, 449–453.

Fan, R.-E., Chen, P.-H. and Lin, C.-J. (2005), 'Working set selection using second order information for training support vector machines', *Journal of Machine Learning Research* **6**, 1889–1918.

Farhangfar, A., Kurgan, L. A. and Pedrycz, W. (2007), 'A novel framework for imputation of missing values in databases', *IEEE Transactions on Systems, Man, and Cybernetics, Part A* **37**(5), 692–709.

Farhangfar, A., Kurgan, L. and Dy, J. (2008), 'Impact of imputation of missing values on classification error for discrete data', *Pattern Recogn.* **41**(12), 3692–3705.

Fayyad, U. and Irani, K. (1993), Multi-interval discretization of continuous-valued attributes for classification learning, *in* '13th International Joint Conference on Uncertainly in Artificial Intelligence(IJCAI93)', pp. 1022–1029.

Feng, H., Guoshun, C., Cheng, Y., Yang, B. and Chen, Y. (2005), A svm regression based approach to filling in missing values, *in* R. Khosla, R. J. Howlett and L. C. Jain, eds, 'KES (3)', Vol. 3683 of *Lecture Notes in Computer Science*, Springer, pp. 581–587.

Frank, E. and Witten, I. (1998), Generating accurate rule sets without global optimization, *in* 'Proceedings of the Fifteenth International Conference on Machine Learning', pp. 144–151.

García-Laencina, P., Sancho-Gómez, J. and Figueiras-Vidal, A. (2009), 'Pattern classification with missing data: a review', *Neural Computation & Applications* **9**(1), 1–12.

García, S. and Herrera, F. (2008), 'An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons', *Journal of Machine Learning Research* **9**, 2677–2694.

Gheyas, I. A. and Smith, L. S. (2010), 'A neural network-based framework for the reconstruction of incomplete data sets', *Neurocomputing* **In Press, Corrected Proof**, –.

Grzymala-Busse, J., Goodwin, L., Grzymala-Busse, W. and Zheng, X. (2005), Handling missing attribute values in preterm birth data sets, *in* '10th International Conference of Rough Sets and Fuzzy Sets and Data Mining and Granular Computing(RSFDGrC05)', pp. 342–351.

Grzymala-Busse, J. W. and Hu, M. (2000), A comparison of several approaches to missing attribute values in data mining., *in* W. Ziarko and Y. Y. Yao, eds, 'Rough Sets and Current Trends in Computing', Vol. 2005 of *Lecture Notes in Computer Science*, Springer, pp. 378–385.

Hruschka, Jr., E. R., Hruschka, E. R. and Ebecken, N. F. (2007), 'Bayesian networks for imputation in classification problems', *J. Intell. Inf. Syst.* **29**(3), 231–252.

Kim, H., Golub, G. H. and Park, H. (2005), 'Missing value estimation for dna microarray gene expression data: local least squares imputation', *Bioinformatics* **21**(2), 187–198.

Kwak, N. and Choi, C.-H. (2002*a*), 'Input feature selection by mutual information based on parzen window', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(12), 1667–1671.

Kwak, N. and Choi, C.-H. (2002*b*), 'Input feature selection for classification problems', *IEEE Transactions on Neural Networks* **13**(1), 143–159.

le Cessie, S. and van Houwelingen, J. (1992), 'Ridge estimators in logistic regression', *Applied Statistics* **41**(1), 191–201.

Little, R. J. A. and Rubin, D. B. (1987), *Statistical Analysis with Missing Data*, Wiley Series in Probability and Statistics, 1st edn, Wiley, New York.

Li, D., Deogun, J., Spaulding, W. and Shuart, B. (2004), Towards missing data imputation: A study of fuzzy k-means clustering method, *in* '4th International Conference of Rough Sets and Current Trends in Computing(RSCTC04)', pp. 573–579.

Luengo, J., García, S. and Herrera, F. (2010), 'A study on the use of imputation methods for experimentation with Radial Basis Function Network classifiers handling missing attribute values: The good synergy between RBFNs and EventCovering method', *Neural Networks* **23**(3), 406–418.

Matsubara, E. T., Prati, R. C., Batista, G. E. A. P. A. and Monard, M. C. (2008), Missing value imputation using a semi-supervised rank aggregation approach, *in* G. Zaverucha and A. C. P. L. da Costa, eds, 'SBIA', Vol. 5249 of *Lecture Notes in Computer Science*, Springer, pp. 217–226.

McLachlan, G. (2004), *Discriminant Analysis and Statistical Pattern Recognition*, John Wiley and Sons.

Merlin, P., Sorjamaa, A., Maillet, B. and Lendasse, A. (2010), 'X-SOM and L-SOM: A double classification approach for missing value imputation', *Neurocomputing* **73**(7-9), 1103–1108.

Michalksi, R., , Mozetic, I. and Lavrac, N. (1986), The multipurpose incremental learning system aq15 and its testing application to three medical domains, *in* '5th INational Conference on Artificial Intelligence ((AAAI86).)', pp. 1041–1045.

Moller, F. (1990), 'A scaled conjugate gradient algorithm for fast supervised learning', *Neural Networks* **6**, 525–533.

Nogueira, B. M., Santos, T. R. A. and Zárate, L. E. (2007), Comparison of classifiers efficiency on missing values recovering: Application in a marketing database with massive missing data., *in* 'CIDM', IEEE, pp. 66–72.

Oba, S., aki Sato, M., Takemasa, I., Monden, M., ichi Matsubara, K. and Ishii, S. (2003), 'A bayesian missing value estimation method for gene expression profile data.', *Bioinformatics* **19**(16), 2088–2096.

Peng, H., Long, F. and Ding, C. (2005), 'Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(8), 1226–1238.

Pham, D. T. and Afify, A. A. (2005), Rules-6: a simple rule induction algorithm for supporting decision making, *in* 'Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE', pp. 2184–2189.

Pham, D. T. and Afify, A. A. (2006), Sri: A scalable rule induction algorithm, *in* 'Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science', Vol. 220, pp. 537–552.

Platt, J. C. (1999), Fast training of support vector machines using sequential minimal optimization, *in* 'Advances in kernel methods: support vector learning', MIT Press, Cambridge, MA, USA, pp. 185–208.

Plat, J. (1991), 'A resource allocating network for function interpolation', *Neural Computation* **3**(2), 213–225.

Pyle, D. (1999), *Data Preparation for Data Mining*, Morgan Kaufmann.

Quinlan, J. (1993), *C4.5: Programs for Machine Learning*, Morgan Kauffman.

Saar-Tsechansky, M. and Provost, F. (2007), 'Handling missing values when applying classification models', *Journal of Machine Learning Research* **8**, 1623–1657.

Schafer, J. L. (1997), *Analysis of Incomplete Multivariate Data*, Chapman & Hall, London.

Schneider, T. (2001), 'Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values', *Journal of Climate* **14**, 853–871.

Song, Q., Shepperd, M., Chen, X. and Liu, J. (2008), 'Can k-NN imputation improve the performance of C4.5 with small software project data sets? a comparative evaluation', *Journal of Systems and Software* **81**(12), 2361–2370.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. and Altman, R. B. (2001), 'Missing value estimation methods for dna microarrays.', *Bioinformatics* **17**(6), 520–525.

Twala, B. (2009), 'An empirical comparison of techniques for handling incomplete data using decision trees', *Applied Artificial Intelligence* **23**, 373–405.

Wang, H. and Wang, S. (2010), 'Mining incomplete survey data through classification', *Knowledge and Information Systems* **24**(2), 221–233.

Wilson, D. (1972), 'Asymptotic properties of nearest neighbor rules using edited data', *IEEE Transactions on Systems and Man and Cybernetics* **2**(3), 408–421.

Wong, A. K. C. and Chiu, D. K. Y. (1987), 'Synthesizing statistical knowledge from incomplete mixed-mode data', *IEEE Trans. Pattern Anal. Mach. Intell.* **9**(6), 796–805.

Wu, X. and Urpani, D. (1999), 'Induction by attribute elimination', *IEEE Transactions on Knowledge and Data Engineering* **11**(5), 805–812.

Zheng, Z. and Webb, G. I. (2000), 'Lazy learning of bayesian rules', *Machine Learning* **41**(1), 53–84.

1.3. Missing data imputation for Fuzzy Rule Based Classification Systems

- J. Luengo, J. Sáez, F. Herrera, Missing data imputation for Fuzzy Rule Based Classification Systems. **Submitted to Soft Computing**.
  - Status: **Submitted**.
  - Impact Factor (JCR 2009): 1.328.
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 51 / 103.
  - Subject Category: Computer Science, Interdisciplinary Applications. Ranking 41 / 95.

Noname manuscript No.
(will be inserted by the editor)

Missing data imputation for Fuzzy Rule Based Classification Systems

Julián Luengo · José A. Sáez · Francisco Herrera

Received: date / Accepted: date

Abstract Fuzzy Rule Based Classification Systems are known due to their ability to treat with low quality data and obtain good results in this scenarios. However, their application in problems with missing data are uncommon while in real-life data, information is frequently incomplete in data mining, caused by the presence of missing values in attributes. Several schemes have been studied to overcome the drawbacks produced by missing values in data mining tasks; one of the most well known is based on preprocessing, formerly known as imputation.

In this work we focus on Fuzzy Rule Based Classification Systems considering fourteen different approaches to Missing attribute Values treatment that are presented and analyzed. The analysis involves a three different methods, in which we distinguish between Mamdani and TSK models. From the obtained results the convenience of using imputation methods for Fuzzy Rule Based Classification Systems with Missing Values is stated. The analysis suggests that each type behaves differently while the use of determined Missing Values imputation methods could improve the accuracy obtained for these methods. Thus the use of particular imputation methods conditioned to the type of Fuzzy Rule Based Classification System is required.

Keywords: Classification; Missing Values; Fuzzy Rule Based Classification Systems; Imputation

Julián Luengo
Dept. of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain
E-mail: julianlm@decsai.ugr.es

José A. Sáez
Dept. of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain
E-mail: smja@decsai.ugr.es

Francisco Herrera
Dept. of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain
E-mail: herrera@decsai.ugr.es

1 Introduction

Many existing, industrial and research data sets contain Missing Values (MVs). There are various reasons for their existence, such as manual data entry procedures, equipment errors and incorrect measurements. The presence of such imperfections requires a preprocessing stage in which the data is prepared and cleaned (Pyle 1999), in order to be useful to and sufficiently clear for the knowledge extraction process. The simplest way of dealing with missing values is to discard the examples that contain them. However, this method is practical only when the data contains a relatively small number of examples with MVs and when analysis of the complete examples will not lead to serious bias during the inference (Little and Rubin 1987).

Fuzzy Rule Based Classification Systems (FRBCSs) (Ishibuchi et al 2004; Kuncheva 2000) are widely employed due to their capability to build a linguistic model interpretable to the users with the possibility of mixing different information. They are also well known for being able to deal with imprecise data. However, few analysis have been carried out considering the presence of MVs (Berthold and Huber 1998; Gabriel and Berthold 2005) for FRBCSs and usually the presence of MVs is not usually taken into account and they are usually discarded, maybe inappropriately. Incomplete data in either the training set or test set or in both sets affect the prediction accuracy of learned classifiers (Gheyas and Smith 2010). The seriousness of this problem depends

in part on the proportion of missing data. Most FRBCSs cannot work directly with incomplete data sets and due to the high dimensionality of real problems it is possible that no valid (complete) cases would be present in the data set (García-Laencina et al 2009).

This inappropriate handling of missing data in the analysis may introduce bias and can result in misleading conclusions being drawn from a research study, and can also limit the generalizability of the research findings (Wang and Wang 2010). Three types of problem are usually associated with missing values in data mining (Barnard and Meng 1999): 1) loss of efficiency; 2) complications in handling and analyzing the data; and 3) bias resulting from differences between missing and complete data.

Therefore the treatment of missing data in data mining is necessary and it can be handled in three different ways normally (Farhangfar et al 2007):

- The first approach is to discard the examples with missing data in their attributes. Therefore deleting attributes with elevated levels of missing data is included in this category too.
- Another approach is the use of maximum likelihood procedures, where the parameters of a model for the complete data are estimated, and later used for imputation by means of sampling.
- Finally, the imputation of MVs is a class of procedures that aims to fill in the MVs with estimated ones. In most cases, a data set's attributes are not independent from each other. Thus, through the identification of relationships among attributes, MVs can be determined.

We will focus our attention on the use of imputation methods. A fundamental advantage of this approach is that the missing data treatment is independent of the learning algorithm used without erasing any example. For this reason, the user can select the most appropriate method for each situation he faces. There is a wide family of imputation methods, from simple imputation techniques like mean substitution, K-Nearest Neighbour, etc.; to those which analyze the relationships between attributes such as: support vector machines-based, clustering-based, logistic regressions, maximum-likelihood procedures and multiple imputation (Batista and Monard 2003; Farhangfar et al 2008).

The literature on imputation methods in Data Mining employs well-known Machine Learning methods for their studies, in which the authors show the convenience of imputing the MVs for the mentioned algorithms, particularly for classification. The vast majority of MVs studies in classification usually analyze and compare one imputation method against a few others under controlled amounts of MVs, and induce them artificially with known mechanisms and probability distributions (Acuna and Rodriguez 2004; Batista and Monard 2003; Farhangfar et al 2008; Jr. et al 2007; Li et al 2004; Luengo et al 2010).

We want to analyze the effect of the use of a large set of imputation methods on FRBCSs, trying to obtain the best imputation procedure for each one. We consider three representative FRBCSs of different nature which have proven to perform well.

- The Fuzzy Hybrid Genetic Based Machine Learning (FH-GBML) method proposed by Ishibuchi et al. (Ishibuchi et al 2005) which is a Mamdani based FRBCS.
- The Fuzzy Rule Learning Model proposed by Chi et al. (Chi) (Chi et al 1996) which is a Mamdani based FRBCSs as well.
- The Positive Definite Fuzzy Classifier (PDFC) proposed by Chen and Wang (Chen and Wang 2003) which is a Takagi-Sugeno (TSK) based FRBCS.

In order to perform the analysis, we use a large bunch of data sets, twenty-one in total, with natural MVs. All the data sets have their proper MVs and we do not induce them, as we want to stay as close to the real world data as possible. First, we analyze the use of the different imputation strategies versus case deletion and the total lack of missing data treatment, for a total of fourteen imputation methods. Therefore, each FRBCS is used over the fourteen imputation results. All the imputation and classification algorithms are publicly available in the KEEL software[1] (Alcalá-Fdez et al 2009). These results are compared using the Wilcoxon Signed Rank test (Demšar 2006; García and Herrera 2008) in order to obtain the best method(s) for each FRBCS. With this information we can extract the best imputation method for each FRBCS, and indicate if there is a common best option depending on the FRBCS type.

We have also analyzed two metrics related to the data characteristics, formerly known as Wilson's noise ratio and Mutual Information. Using these measures, we have observed the influence of the imputation procedures on the noise and on the relationship of the attributes with the class label as well. This procedure tries to quantify the quality of each imputation method independently of the classification algorithm.

The rest of the paper is organized as follows. Section 2 introduces the descriptions of the FRBCSs considered and a brief review of the current state of the art in MVs for FRBCSs. In Section 3 we present the basis of the application of the imputation methods and the description of the imputation methods we have used. In Sec-

[1] `http://keel.es`

tion 4, the experimental framework, the classification methods and the parameters used for both imputation and classification methods are presented. In Section 5, the results obtained are analyzed. In Section 6 we use two measures to quantify the influence of the imputation methods in the data sets, both in the instances and in the features. Finally, in Section 7 we make some concluding remarks.

2 Fuzzy Rule Based Classification Systems

In this section we describe the basis of the three models that we have used in our study. First we introduce the basic notation that we will use later. Next we describe the Chi method (Subsection 2.1), the FH-GBML method (Subsection 2.2) and the PDFC method (Subsection 2.3). In Subsection 2.4 we describe the contributions made to the MVs treatment for FRBCSs and we tackle the different situations which apply for the three FRBCSs considered when MVs appear.

Any classification problem consists of w training patterns $x_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, m$ from M classes where x_{pi} is the ith attribute value ($i = 1, 2, \ldots, n$) of the p-th training pattern.

In this work we use fuzzy rules in the following form:

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A_j^1 \text{ and } \ldots \text{ and } x_n \text{ is } A_j^n \text{ then Class} = C_j \text{ with } RW_j \quad (1)$$

where R_j is the label of the jth rule, $x = (x_1, \ldots, x_n)$ is an n-dimensional pattern vector, A_j^i is an antecedent fuzzy set, C_j is a class label or a numeric value, and RW_j is the rule weight. We always use triangular membership functions as antecedent fuzzy sets.

2.1 Chi et al. Approach

This FRBCSs design method (Chi et al 1996) is an extension of the well-known Wang and Mendel method (Wang and Mendel 1992) for classification problems. To generate the fuzzy Rule Base (RB), it determines the relationship between the variables of the problem and establishes an association between the space of the features and the space of the classes by means of the following steps:

Step 1: *Establishment of the linguistic partitions.* Once the domain of variation of each feature A_i is determined, the fuzzy partitions are computed.

Step 2: *Generation of a fuzzy rule for each example* $x_p = (x_{p1}, \ldots, x_{pn}, C_p)$. To do this it is necessary:

Step 2.1: To compute the matching degree $\mu(x_p)$ of the example to the different fuzzy regions using a conjunction operator (usually modeled with a minimum or product T-norm).

Step 2.2: To assign the example x_p to the fuzzy region with the greatest membership degree.

Step 2.3: To generate a rule for the example, whose antecedent is determined by the selected fuzzy region and whose consequent is the label of class of the example.

Step 2.4: To compute the rule weight.

We must remark that rules with the same antecedent can be generated during the learning process. If they have the same class in the consequent we just remove one of the duplicated rules, but if they have a different class only the rule with the highest weight is kept in the RB.

2.2 Fuzzy Hybrid Genetic Based Machine Learning Rule Generation Algorithm

The basis of the algorithm described here (Ishibuchi et al 2005), consists of a Pittsburgh approach where each rule set is handled as an individual. It also contains a Genetic Cooperative-Competitive Learning (GCCL) approach (an individual represents a unique rule), which is used as a kind of heuristic mutation for partially modifying each rule set, because of its high search ability to efficiently find good fuzzy rules.

The system defines 14 possible linguistic terms for each attribute, as shown in Figure 1, which correspond to Ruspini's strong fuzzy partitions with two, three, four, and five uniformly distributed triangular-shaped membership functions. Furthermore, the system also uses "don't care" as an additional linguistic term, which indicates that the variable matches any input value with maximum matching degree.

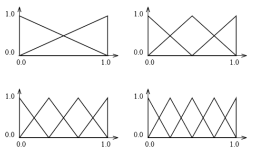

Fig. 1 Four fuzzy partitions for each attribute membership function

The main steps of this algorithm are described below:

Step 1: Generate N_{pop} rule sets with N_{rule} fuzzy rules.

Step 2: Calculate the fitness value of each rule set in the current population.

Step 3: Generate (N_{pop} -1) rule sets by selection, crossover and mutation in the same manner as the Pittsburgh-style algorithm. Apply a single iteration of the GCCL-style algorithm (i.e., the rule generation and the replacement) to each of the generated rule sets with a pre-specified probability.

Step 4: Add the best rule set in the current population to the newly generated (N_{pop} -1) rule sets to form the next population.

Step 5: Return to Step 2 if the pre-specified stopping condition is not satisfied.

Next, we will describe every process of the algorithm:

- Initialization: N_{rule} training patterns are randomly selected. Then, a fuzzy rule from each of the selected training patterns is generated by choosing probabilistically (as shown in (2)) an antecedent fuzzy set from the 14 candidates $B_k(k = 1, 2, \ldots, 14)$ (see Figure 1) for each attribute. Then each antecedent fuzzy set of the generated fuzzy rule is replaced with *don't care* using a pre-specified probability $P_{don't\ care}$.

$$P_{don't\ care}(B_k) = \frac{\mu_{B_k}(x_{pi})}{\sum_{j=1}^{14} \mu_{B_j}(x_{pi})} \tag{2}$$

- Fitness computation: The fitness value of each rule set S_i in the current population is calculated as the number of correctly classified training patterns by S_i. For the GCCL approach the computation follows the same scheme.
- Selection: It is based on binary tournament.
- Crossover: The substring-wise and bit-wise uniform crossover are applied in the Pittsburgh part. In the case of the GCCL part only the bit-wise uniform crossover is considered.
- Mutation: Each fuzzy partition of the individuals is randomly replaced with a different fuzzy partition using a pre-specified mutation probability for both approaches.

2.3 Positive Definite Function Classifier

The PDFC learning method (Chen and Wang 2003) uses a Support Vector Machine (SVM) approach to build up the model. PDFC considers a fuzzy model with $m + 1$ fuzzy rules of the form given in Equation (1) where A_j^k is a fuzzy set with membership function $a_j^k : \mathbb{R} \to [0, 1]$, $RW_j = 1$ and $C_j = b_j \in \mathbb{R}$. Therefore PDFC is a FRBCS with constant `THEN`-parts. If we choose product as the fuzzy conjunction operator, addition for fuzzy rule aggregation and center of area defuzzification, then the model becomes a special form of the Takagi-Sugeno fuzzy model.

PDFC considers the use of membership functions generated from a reference function a^k through location transformation (Dubois and Prade 1978). In (Chen and Wang 2003) well-known types of reference functions can be found, like the symmetric triangle and the gaussian function. As a consequence of the presented formulation,

$$K(x_p, z_j) = \prod_{k=1}^{n} a^k(x_p^k - z_j^k) \tag{3}$$

is a Mercer Kernel (Cristianini and Shawe-Taylor 2000), if it has nonnegative Fourier transform. Thus, the decision rule of a binary fuzzy classifier is

$$f(x_p) = \text{sign}\left(b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_p^k)\right). \tag{4}$$

So the remaining question is how to find a set of fuzzy rules ($\{z_1, \ldots, z_m\}$ and $\{b_0, \ldots, b_m\}$). It is well-known that the SVM algorithm finds a separating hyperplane with good generalization by reducing the empirical risk and, at the same time, controlling the hyperplane margin (Vapnik 1998). Thus we can use the SVM algorithm to find an optimal hyperplane in $\mathbb{F}$. Once we get such a hyperplane, fuzzy rules can easily be extracted. The whole procedure is described next:

Step 1: Construct a Mercer kernel, K, from the given positive-definite reference functions according to (3).

Step 2: Construct an SVM to get a decision rule of the form

$$f(x) = sign\left(\sum_{i \in S} y_i \alpha_i K(x, x_i) + b\right),$$

with S as the index set of the support vectors:

Step 2.1: Assign some positive number to the cost C, and solve the quadratic program defined by the proper SVM to get the Lagrange multipliers α_i.

Step 2.2: Find b (details can be found in, for example, (Platt 1999)).

Step 3: Extract fuzzy rules from the decision rule of the SVM:

Step 3.1: b_0 is the constant parameter of the hyperplane, that is $b_0 \leftarrow b$

Step 3.2: For each support vector create a fuzzy rule where: we center the reference functions on the support vector $z_j \leftarrow x_i$ and we assign the rule consequent $b_j \leftarrow y_i \alpha_i$

2.4 Missing values treatment for Fuzzy Rule Based Classification Systems

Traditionally, the presence of MVs in the data has not been considered when building up the FRBCS model.

Although the FRBCS are capable of managing imperfect data, their abilities has not been explicitly checked in this case. The only precedent in the literature of FRBCSs learning in the case of MVs is a technique proposed to tolerate MVs in the training of a FRBCS by Berthold and Huber (1998). This procedure was initially intended to estimate the best approximation to the MV based on the core region of the fuzzy label associated to the missing attribute.

This initial work was further developed applying the initial technique to a particular fuzzy rule induction algorithm in (Gabriel and Berthold 2005). The main idea was to avoid the use of the missing attribute in the rule operations when covering new examples or specializing the rules. This is a simple and easy to implement idea, but its extension is limited to few fuzzy rule induction algorithms, like FH-GBML.

As we can appreciate from the mentioned studies there is a lack of research in this area. There are many different approaches to the treatment of MVs, which use many different methods (to classify and to impute MVs), but they have not been considered with FRBCSs. Therefore, in spite of the variety of studies presented, the necessity of analyze the use of imputation methods for FRBCSs is demonstrated.

Only one of the presented classification methods in the previous section have its own MVs treatment. We have applied the procedure indicated in (Gabriel and Berthold 2005) using the "don't care" label, but this extension is not easy to apply to Chi et al. and PDFC algorithms due to their different nature. For this reason PDFC and Chi et al. FRBCSs are not able to deal with MVs. Thus we set the training and test accuracy to zero in the presence of MVs, as the methods cannot build a model or compute a distance to the instance.

3 Imputation background

In this section we first set the basis of our study in accordance with the MV literature. The rest of this section is organized as follows: In Subsection 3.1 we indicate the fundamental aspects in the MVs treatment based on the MV introduction mechanism. In Subsection 3.2 we have summarized the imputation methods that we have used in our study.

A more extensive and detailed description of these methods can be found on the web page `http://sci2s.ugr.es/MVDM`, and a PDF file with the original source paper descriptions is present on the web page formerly named "Imputation of Missing Values. Methods' Description". A more complete bibliography section is also available on the mentioned web page.

3.1 Missing values introduction mechanisms

It is important to categorize the mechanisms which lead to the introduction of MVs (Little and Rubin 1987). The assumptions we make about the missingness mechanism and the missing data pattern of missing values can affect which imputation method could be applied, if any. As Little and Rubin (1987) stated, there are three different mechanisms for missing data induction:

1. Missing completely at random (**MCAR**), when the distribution of an example having a missing value for an attribute does not depend on either the observed data or the missing data.
2. Missing at random (**MAR**), when the distribution of an example having a missing value for an attribute depends on the observed data, but does not depend on the missing data.
3. Not missing at random (**NMAR**), when the distribution of an example having a missing value for an attribute depends on the missing values.

In the case of the MCAR mode, the assumption is that the underlying distributions of missing and complete data are the same, while for the MAR mode they are different, and the missing data can be predicted by using the complete data (Little and Rubin 1987). These two mechanisms are assumed by the imputation methods so far. As Farhangfar et al (2008) and Matsubara et al (2008) state, it is only in the MCAR mechanism case where the analysis of the remaining complete data (ignoring the incomplete data) could give a valid inference (classification in our case) due to the assumption of equal distributions. That is, case and attribute removal with missing data should be applied only if the missing data is MCAR, as both of the other mechanisms could potentially lead to information loss that would lead to the generation of a biased/incorrect classifier (i.e. a classifier based on a different distribution).

Another approach is to convert the missing values to a new value (encode them into a new numerical value), but such a simplistic method was shown to lead to serious inference problems (Schafer 1997). On the other hand, if a significant number of examples contain missing values for a relatively small number of attributes, it may be beneficial to perform imputation (filling-in) of the missing values. In order to do so, the assumption of MAR randomness is needed, as Little and Rubin (1987) observed in their analysis.

In our case we will use single imputation methods, due to the time complexity of the multiple imputation schemes, and the assumptions they make regarding data distribution and MV randomness; that is, that we should know the underlying distributions of the complete data and missing data prior to their application.

3.2 Description of the imputation methods

In this subsection, we briefly describe the imputation methods that we have used.

- Do Not Impute (**DNI**). As its name indicates, all the missing data remains unreplaced, so the FRBCSs must use their default MVs strategies. The objective is to verify whether imputation methods allow the classification methods to perform better than when using the original data sets. As a guideline, in Grzymala-Busse and Hu (2000) a previous study of imputation methods is presented.
- Case deletion or Ignore Missing (**IM**). Using this method, all instances with at least one MV are discarded from the data set.
- Global Most Common Attribute Value for Symbolic Attributes, and Global Average Value for Numerical Attributes (**MC**)(Grzymala-Busse et al 2005). This method is very simple: for nominal attributes, the MV is replaced with the most common attribute value, and numerical values are replaced with the average of all values of the corresponding attribute.
- Concept Most Common Attribute Value for Symbolic Attributes, and Concept Average Value for Numerical Attributes (**CMC**)(Grzymala-Busse et al 2005). As stated in *MC*, the MV is replaced by the most repeated one if nominal or the mean value if numerical, but considering only the instances with the same class as the reference instance.
- Imputation with K-Nearest Neighbor (**KNNI**) (Batista and Monard 2003). Using this instance-based algorithm, every time an MV is found in a current instance, KNNI computes the k nearest neighbors and a value from them is imputed. For nominal values, the most common value among all neighbors is taken, and for numerical values the average value is used. Therefore, a proximity measure between instances is needed for it to be defined. The euclidean distance (it is a case of a L_p norm distance) is the most commonly used in the literature.
- Weighted imputation with K-Nearest Neighbor (**WKNNI**)(Troyanskaya et al 2001). The Weighted K-Nearest Neighbor method selects the instances with similar values (in terms of distance) to a considered one, so it can impute as *KNNI* does. However, the estimated value now takes into account the different distances from the neighbors, using a weighted mean or the most repeated value according to the distance.
- K-means Clustering Imputation (**KMI**)(Li et al 2004). Given a set of objects, the overall objective of clustering is to divide the data set into groups based on the similarity of objects, and to minimize the intra-cluster dissimilarity. KMI measures the intra-cluster dissimilarity by the addition of distances among the objects and the centroid of the cluster which they are assigned to. A cluster centroid represents the mean value of the objects in the cluster. Once the clusters have converged, the last process is to fill in all the non-reference attributes for each incomplete object based on the cluster information. Data objects that belong to the same cluster are taken to be nearest neighbors of each other, and KMI applies a nearest neighbor algorithm to replace missing data, in a similar way to KNNI.
- Imputation with Fuzzy K-means Clustering (**FKMI**) (Acuna and Rodriguez 2004; Li et al 2004). In fuzzy clustering, each data object has a membership function which describes the degree to which this data object belongs to a certain cluster. In the process of updating membership functions and centroids, FKMI's only take into account complete attributes. In this process, the data object cannot be assigned to a concrete cluster represented by a cluster centroid (as is done in the basic K-mean clustering algorithm), because each data object belongs to all K clusters with different membership degrees. FKMI replaces non-reference attributes for each incomplete data object based on the information about membership degrees and the values of cluster centroids.
- Support Vector Machines Imputation (**SVMI**)(Feng et al 2005) is an SVM regression based algorithm to fill in missing data, i.e. set the decision attributes (output or classes) as the condition attributes (input attributes) and the condition attributes as the decision attributes, so SVM regression can be used to predict the missing condition attribute values. In order to do that, first SVMI selects the examples in which there are no missing attribute values. In the next step the method sets one of the condition attributes (input attribute), some of those values that are missing, as the decision attribute (output attribute), and the decision attributes as the condition attributes by contraries. Finally, an SVM regression is used to predict the decision attribute values.
- Event Covering (**EC**)(Wong and Chiu 1987). Based on the work of Wong and Chiu (1987), a mixed-mode probability model is approximated by a discrete one. First, EC discretizes the continuous components using a minimum loss of information criterion. Treating a mixed-mode feature n-tuple as a discrete-valued one, a new statistical approach is proposed for the synthesis of knowledge based on cluster analysis. The main advantage of this method is that it does not require either scale normalization or the ordering of discrete values. By synthesizing

the data into statistical knowledge, the EC method involves the following processes: 1) synthesize and detect from data inherent patterns which indicate statistical interdependency; 2) group the given data into inherent clusters based on this detected interdependency; and 3) interpret the underlying patterns for each cluster identified. The method of synthesis is based on the author's *event–covering* approach. With the developed inference method, EC is able to estimate the MVs in the data.

- Regularized Expectation-Maximization (**EM**) (Schneider 2001). Missing values are imputed with a regularized expectation maximization (EM) algorithm. In an iteration of the EM algorithm, given estimates of the mean and of the covariance matrix are revised in three steps. First, for each record with missing values, the regression parameters of the variables with missing values among the variables with available values are computed from the estimates of the mean and of the covariance matrix. Second, the missing values in a record are filled in with their conditional expectation values given the available values and the estimates of the mean and of the covariance matrix, the conditional expectation values being the product of the available values and the estimated regression coefficients. Third, the mean and the covariance matrix are re-estimated, the mean as the sample mean of the completed data set and the covariance matrix as the sum of the sample covariance matrix of the completed data set and an estimate of the conditional covariance matrix of the imputation error. The EM algorithm starts with initial estimates of the mean and of the covariance matrix and cycles through these steps until the imputed values and the estimates of the mean and of the covariance matrix stop changing appreciably from one iteration to the next.
- Singular Value Decomposition Imputation (**SVDI**) (Troyanskaya et al 2001). In this method, singular value decomposition is used to obtain a set of mutually orthogonal expression patterns that can be linearly combined to approximate the values of all attributes in the data set. In order to do that, first SVDI estimates the MVs within the *EM* algorithm, and then it computes the Singular Value Decomposition and obtains the eigenvalues. Now SVDI can use the eigenvalues to apply a regression to the complete attributes of the instance, to obtain an estimation of the MV itself.
- Bayesian Principal Component Analysis(**BPCA**) (Oba et al 2003). This method is an estimation method for missing values, which is based on Bayesian principal component analysis. Although the methodology that a probabilistic model and latent variables are estimated simultaneously within the framework of Bayesian inference is not new in principle, actual BPCA implementation that makes it possible to estimate arbitrary missing variables is new in terms of statistical methodology. The missing value estimation method based on BPCA consists of three elementary processes. They are (1) principal component (PC) regression, (2) Bayesian estimation, and (3) an expectationmaximization (EM)-like repetitive algorithm.
- Local Least Squares Imputation (**LLSI**)(Kim et al 2005). With this method, a target instance that has missing values is represented as a linear combination of similar instances. Rather than using all available genes in the data, only similar genes based on a similarity measure are used. The method has the "local" connotation. There are two steps in the LLSI. The first step is to select k genes by the L_2-norm. The second step is regression and estimation, regardless of how the k genes are selected. A heuristic k parameter selection method is used by the authors.

4 Experimental framework

When analyzing imputation methods, a wide range of set ups can be observed. The data sets used, their type (real or synthetic), the origin and amount of MVs, etc. must be carefully described, as the results will strongly depend on them. All these aspects are described in Subsection 4.1.

The results obtained by the classification methods depend on the previous imputation step, but also on the parameter configuration used by both the imputation and classification methods. Therefore they must be indicated in order to be able to reproduce any results obtained. In Subsection 4.2 the parameter configurations used by all the methods considered in this study are presented.

4.1 Data sets description

The experimentation has been carried out using 21 benchmark data sets from the KEEL-Dataset repository[2]. Each data set is described by a set of characteristics such as the number of data samples, attributes and classes, summarized in Table 1. In this table, the percentage of MVs is indicated as well: the percentage of values which are missing, and the percentage of instances with at least one MV.

[2] `http://sci2s.ugr.es/keel/datasets.php`

Table 1 Data sets used

Data set	Acronym	# instances.	# attributes	# classes	% MV	% inst. with MV
Cleveland	CLE	303	14	5	0.14	1.98
Wisconsin	WIS	699	10	2	0.23	2.29
Credit	CRX	689	16	2	0.61	5.37
Breast	BRE	286	10	2	0.31	3.15
Autos	AUT	205	26	6	1.11	22.44
Primary tumor	PRT	339	18	21	3.69	61.06
Dermatology	DER	365	35	6	0.06	2.19
House-votes-84	HOV	434	17	2	5.3	46.54
Water-treatment	WAT	526	39	13	2.84	27.76
Sponge	SPO	76	46	12	0.63	28.95
Bands	BAN	540	40	2	4.63	48.7
Horse-colic	HOC	368	24	2	21.82	98.1
Audiology	AUD	226	71	24	1.98	98.23
Lung-cancer	LUN	32	57	3	0.27	15.63
Hepatitis	HEP	155	20	2	5.39	48.39
Mushroom	MUS	8124	23	2	1.33	30.53
Post-operative	POS	90	9	3	0.37	3.33
Echocardiogram	ECH	132	12	4	4.73	34.09
Soybean	SOY	307	36	19	6.44	13.36
Mammographic	MAM	961	6	2	2.81	13.63
Ozone	OZO	2534	73	2	8.07	27.11

We cannot know anything about the randomness of MVs in the data sets, so we assume they are distributed in an *MAR* way, so the application of the imputation methods is feasible. In our study we want to deal with the original MVs and therefore obtain the real accuracy values of each data set with our imputation methods. In addition to this, we use all kinds of data sets, which includes nominal data sets, numeric data sets and mixed-mode data sets.

In order to carry out the experimentation, we have used a 10-fold cross validation scheme. All the classification algorithms use the same partitions, to perform fair comparisons. We take the mean accuracy of training and test of the 10 partitions as a representative measure of the method's performance.

All these data sets have natural MVs, and we have imputed them with the following scheme. With the training partition, we apply the imputation method, extracting the relationships between the attributes, and filling in this partition. Next, with the information obtained, we fill in the MVs in the test partition. Since we have 14 imputation methods, we will obtain 14 instances of each partition of a given data set once they have been preprocessed. All these partitions will be used to train the classification methods used in our study, and then we will perform the test validation with the corresponding test partition. If the imputation method works only with numerical data, the nominal values are considered as a list of integer values, starting from 1 to the amount of different nominal values in the attribute.

4.2 Parameter configuration

In Table 2 we show the parameters used by each imputation method described in Section 3.2, in cases where the method needs a parameter. The values chosen are those recommended by their respective authors. Please refer to their respective papers for further descriptions of the parameters' meaning.

Table 2 Imputation Methods Parameters

Method	Parameters
SVMI	Kernel= RBF C= 1.0 Epsilon= 0.001 shrinking= No
KNNI, WKNNI	K= 10
KMI	K= 10 iterations = 100 error = 100
FKMI	K= 3 iterations = 100 error = 100 m = 1.5
EC	T= 0.05
EM	iterations = 30 stagnation tolerance = 0.0001 inflation factor = 1 regression type = multiple ridge regression
SVDI	iterations = 30 stagnation tolerance = 0.005 inflation factor = 1 regression type = multiple ridge regression singular vectors = 10
LLSI	max number of nearest neighbor = 200

In Table 3 the parameters used by the different FRBCSs are presented. All these parameters are the recommended ones that have been extracted from the respective publications of the methods. Please refer to the associated publications and the KEEL platform to obtain further details about the meaning of the different parameters.

Table 3 Parameters used by the FRBCSs (p is the number of attributes in the data set)

FH-GBML
Number of fuzzy rules: $5 \times p$ rules.
Number of rule sets (N_{pop}): 200 rule sets.
Crossover probability: 0.9.
Mutation probability: $1/p$.
Number of replaced rules: All rules except the best-one (Pittsburgh-part, elitist approach), number of rules/5 (Michigan-part).
Total number of generations: 1,000 generations.
Don't care probability: 0.5.
Probability of the application of the Michigan iteration: 0.5
PDFC
C =100
d = 0.25
Positive definite function type: symmetric triangle
Chi
Number of Labels: 3
T-norm: product
Rule weight: penalized certainty factor
Fuzzy reasoning method: winning rule

5 Analysis of the imputation methods for Fuzzy Rule Based Classification Systems

In this section we analyze the imputation results obtained for the FRBCSs and study the best imputation choices in each case. We first show the test accuracy results for the three FRBCSs using the 14 imputation methods in Subsection 5.1 and indicate the best approaches using this initial criteria. In order to establish a more robust and significant comparison we have used the Wilcoxon Signed Rank test in Subsection 5.2, to support our analysis with a statistical test that provides us with statistical evidence of the good behavior of any imputation approach for the FRBCSs.

5.1 Results for all the classification methods

In this section, we analyze the different imputation approaches for all the imputation methods as a first attempt to obtain an "overall best" imputation method for each FRBCS. Following the indications given in the previous subsection, in Table 4 we depict the average test accuracy for the three FRBCSs for each imputation method and data set. The best imputation method in each case is stressed in bold face. We include a final column with the average accuracy across all data sets for each imputation method.

Attending to the average test accuracy obtained, the best imputation methods are:

- DNI for FH-GBML. The use of the "don't care" option when MVs appear obtains good results in comparison with the rest of imputation methods. This is specially appreciable in the case of HOC and AUD data sets. The CMC method is also close in the final average, while it presents less differences with the best method when it is not the best one unlike DNI.
- BPCA for Chi et al. Although BPCA presents an irregular behavior, its superior performance in DER, HOV, BAN, LUN and HEP data sets allows it to obtain a better average. Again, CMC is the second best method and its behavior is more similar to the rest of the methods with less variations.
- EC for PDFC. The results for PDFC are less irregular, and EC is consistently better in the majority of them. In contraposition with FH-GBML and Chi et al., in this case the best imputation method obtains a clear difference with the second best, SVMI in this case.

From these results an initial recommendation of the best imputation procedures for each FRBCS can be made. However, the high variations in the results discourages to use the accuracy as the criteria to select them, specially for FH-GBML and Chi et al. methods. Therefore a more robust procedure must be used in the comparisons in order to obtain the best imputation method for each FRBCS. This is discussed in the next subsection.

5.2 Statistical analysis

In order to appropriately analyze the imputation and classification methods, we apply the Wilcoxon Signed rank test comparing the imputation methods for each FRBCS separately. With the results of the test we create one table per FRBCS in which we provide an average ranking for each imputation method indicating the best ones. The content of the tables and its interpretation is as follows:

1. We create an $n \times n$ table for each classification method. In each cell, the outcome of the Wilcoxon signed rank test is shown.
2. In the aforementioned tables, if the p-value obtained by the Wilcoxon tests for a pair of imputation methods is higher than our α level, formerly 0.1, then we establish that there is a *tie* in the comparison (no significant difference was found), represented by a D.
3. If the p-value obtained by the Wilcoxon tests is lower than our α level, formerly 0.1, then we establish that there is a *win* (represented by a W) or a *loss* (represented by an L) in the comparison. If the method presented in the row has a better ranking than the method presented in the column in the Wilcoxon test then there is a *win*, otherwise there is a *loss*.

Table 4 Test accuracy results for FH-GBML

FRBCS	Imputation Method	CLE	WIS	CRX	BRE	AUT	PRT	DER	HOV	WAT	SPO	BAN	HOC	AUD	LUN	HEP	MUS	POS	ECH	SOY	MAM	OZO	Average
FH-GBML	IM	52.21	**96.92**	85.25	70.79	**50.31**	21.61	29.47	91.79	**67.70**	16.38	65.59	0.00	20.00	0.00	81.82	**85.81**	62.36	52.39	18.48	83.18	84.91	54.14
	EC	52.81	94.57	85.21	74.10	33.74	28.57	26.58	93.77	28.35	19.29	**67.41**	76.64	36.34	0.00	75.50	80.09	64.44	45.44	17.62	82.63	5.96	51.86
	KNNI	49.82	95.86	84.19	69.22	41.72	21.43	**33.96**	93.53	62.55	16.96	57.78	49.89	18.68	0.00	74.92	81.43	**68.89**	59.89	12.33	83.67	86.94	55.41
	WKNNI	53.11	96.14	83.76	71.08	43.73	21.60	21.40	92.63	64.65	8.93	58.15	45.78	11.11	0.00	78.08	81.30	65.56	59.07	15.62	82.73	83.35	54.18
	KMI	53.78	96.29	84.48	72.75	45.91	23.78	21.99	92.85	66.54	6.61	55.93	71.43	20.59	0.00	77.38	80.70	61.11	53.63	19.48	83.46	84.37	55.86
	FKMI	52.75	96.71	84.63	70.94	41.60	20.99	22.79	92.61	60.85	14.11	62.78	76.34	19.86	0.00	78.75	81.65	63.33	52.97	**26.39**	83.67	81.49	56.44
	SVMI	54.48	96.43	84.48	75.51	42.58	25.00	31.28	93.77	64.82	14.82	57.78	45.29	20.47	0.00	81.83	80.11	63.33	52.20	17.85	82.94	86.74	55.80
	EM	51.83	96.14	84.91	73.11	45.31	8.43	27.65	86.87	54.96	14.46	53.15	75.73	0.00	0.00	77.46	83.15	65.56	56.15	10.42	82.62	76.95	53.57
	SVDI	52.82	96.42	83.47	73.07	45.89	14.76	27.91	89.17	57.82	13.04	50.19	77.45	0.00	0.00	**83.17**	78.06	66.67	50.60	14.74	81.69	74.50	53.88
	BPCA	48.49	43.78	47.31	**78.03**	49.37	**34.26**	8.24	49.97	46.75	2.50	52.78	56.28	12.77	**25.00**	78.08	43.00	51.11	36.37	0.00	51.19	80.67	42.66
	LLSI	**56.08**	96.57	83.76	72.71	44.71	24.15	24.92	**94.93**	61.61	14.46	55.74	63.83	29.58	0.00	73.33	83.68	63.33	57.69	14.35	83.15	83.39	56.28
	MC	53.45	96.43	84.34	72.71	42.48	25.43	30.40	91.46	64.83	14.64	59.81	76.30	21.76	0.00	78.04	81.43	62.22	50.00	15.66	83.46	84.10	56.62
	CMC	53.77	96.43	**85.65**	71.03	46.11	25.36	31.53	94.01	64.28	16.96	55.56	70.17	14.64	0.00	78.63	82.62	63.33	**61.59**	15.04	**83.67**	**86.94**	57.02
	DNI	54.45	96.14	84.77	73.48	42.21	23.36	26.00	94.70	55.92	**21.79**	62.59	**84.77**	**42.02**	0.00	74.88	75.70	63.33	52.20	24.04	81.90	65.03	**57.11**
Chi et al.	IM	39.69	91.51	**81.77**	66.88	61.99	9.97	5.07	42.25	64.42	**14.95**	35.69	0.00	**20.00**	0.00	18.70	99.47	23.19	**61.92**	10.20	78.89	91.46	43.71
	EC	**44.17**	75.11	50.09	65.09	48.80	10.27	4.93	46.55	1.90	10.54	37.22	6.57	5.38	0.00	23.25	99.61	22.22	49.18	9.11	78.98	2.92	32.95
	KNNI	39.89	91.42	80.71	65.45	59.51	11.38	4.93	41.93	64.66	11.79	38.52	1.35	0.91	0.00	30.96	93.61	22.22	58.96	9.43	78.67	92.70	42.81
	WKNNI	39.89	91.42	80.71	65.45	59.51	11.38	4.93	41.93	64.66	11.79	38.52	1.35	0.91	0.00	30.96	93.61	22.22	58.96	9.43	78.67	92.78	42.81
	KMI	39.89	91.42	81.14	65.45	59.55	12.70	4.93	41.49	65.03	11.79	29.44	4.06	5.34	0.00	22.29	**99.74**	22.22	54.56	10.40	78.25	92.26	42.47
	FKMI	39.89	91.28	81.14	65.45	59.55	12.38	4.93	42.17	64.07	11.79	37.22	5.17	0.91	0.00	30.96	**99.74**	22.22	56.04	**11.70**	78.04	91.91	43.17
	SVMI	40.23	**91.56**	80.85	65.45	58.93	9.34	4.93	41.93	64.65	11.79	38.52	1.62	5.45	0.00	30.29	**99.74**	22.22	60.38	10.42	80.85	92.78	43.43
	EM	39.88	91.13	80.71	66.12	68.82	4.93	4.93	44.47	62.00	10.54	29.63	29.37	0.91	0.00	25.13	99.29	22.22	58.41	8.78	79.61	91.95	43.75
	SVDI	40.88	91.28	80.56	66.12	67.36	5.86	4.93	45.62	58.96	10.54	27.78	26.92	0.91	0.00	24.46	96.55	22.22	56.92	8.78	78.77	91.59	43.19
	BPCA	39.60	44.78	49.49	**78.42**	**82.21**	**67.03**	**19.50**	**50.20**	50.19	3.75	**56.11**	**54.37**	18.12	**28.33**	**73.54**	50.84	**42.22**	36.37	3.26	51.92	87.41	**47.03**
	LLSI	39.89	91.28	80.85	65.09	54.76	8.29	4.93	47.45	59.71	11.96	32.96	2.17	12.39	0.00	31.58	97.38	22.22	55.16	8.78	79.61	92.34	42.80
	MC	39.89	91.28	81.00	65.45	57.90	12.38	4.93	42.17	64.27	11.79	38.70	2.47	0.91	0.00	31.58	**99.74**	22.22	59.07	**11.70**	78.04	91.99	43.21
	CMC	39.89	91.42	80.71	65.45	60.40	10.74	4.93	47.93	**65.79**	11.79	37.96	11.98	0.91	0.00	31.58	**99.74**	22.22	59.73	10.41	**81.06**	**92.82**	44.16
	DNI	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PDFC	IM	52.54	**96.48**	83.40	73.65	80.96	42.46	88.84	94.92	74.39	70.05	75.39	40.00	20.00	36.67	**83.25**	**100.00**	52.50	56.54	91.26	83.13	92.65	70.91
	EC	**58.43**	96.28	83.03	72.36	85.61	42.70	88.81	94.47	73.78	**72.68**	**79.07**	82.62	75.36	**40.00**	82.58	**100.00**	55.56	**63.57**	91.24	82.62	93.49	**76.87**
	KNNI	53.10	95.57	**83.61**	71.63	76.71	43.38	88.81	94.23	76.65	**72.68**	77.22	80.99	75.73	**40.00**	77.50	**100.00**	54.44	52.69	89.28	82.94	93.77	75.28
	WKNNI	53.10	95.57	**83.61**	71.63	76.71	43.02	88.81	94.23	77.03	**72.68**	77.22	81.26	75.73	**40.00**	77.50	**100.00**	53.33	51.92	89.60	82.94	93.77	75.22
	KMI	52.12	95.71	82.60	69.88	76.23	43.73	88.81	94.69	76.27	**72.68**	76.85	83.43	**76.66**	**40.00**	79.46	**100.00**	55.56	53.24	91.24	82.52	94.16	75.52
	FKMI	52.12	95.57	82.74	69.88	76.71	43.40	88.81	94.93	75.90	**72.68**	77.78	83.45	76.23	**40.00**	80.04	**100.00**	54.44	57.09	91.24	82.94	93.80	75.70
	SVMI	53.12	95.42	83.46	69.88	77.64	41.83	88.81	93.78	**78.36**	**72.68**	78.89	80.98	76.23	**40.00**	82.58	**100.00**	**56.67**	54.84	90.91	82.73	94.04	75.85
	EM	53.43	95.57	83.17	71.26	83.05	31.04	**89.08**	90.12	70.01	**72.68**	74.63	77.65	22.61	**40.00**	81.38	**100.00**	54.44	58.52	88.62	83.04	93.25	72.07
	SVDI	53.13	95.71	82.74	72.66	83.52	32.33	**89.08**	90.57	70.58	**72.68**	74.63	77.93	24.86	**40.00**	79.46	**100.00**	54.44	55.44	88.62	82.31	93.37	72.10
	BPCA	38.94	43.92	53.85	**93.10**	**90.58**	**69.78**	19.50	50.20	50.37	3.75	57.22	54.37	18.12	28.33	73.54	50.84	43.33	36.37	3.26	50.36	85.00	48.32
	LLSI	52.43	95.57	83.18	69.53	77.68	38.38	88.81	94.70	73.80	**72.68**	78.33	82.34	72.71	**40.00**	78.71	**100.00**	53.33	53.96	86.69	82.83	93.73	74.73
	MC	53.11	95.71	82.75	69.88	75.84	43.46	88.81	94.93	75.69	**72.68**	77.22	82.91	76.23	**40.00**	77.50	**100.00**	55.56	55.55	91.24	82.94	93.37	75.49
	CMC	52.10	95.42	83.03	69.88	76.71	41.71	88.81	**95.16**	77.80	**72.68**	73.70	**85.89**	75.36	**40.00**	80.71	**100.00**	54.44	54.07	**91.88**	**83.77**	**94.20**	75.59
	DNI	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

With these columns, we have produced an average ranking for each FRBCS. We have computed the number of times that an imputation methods wins, and the number of times that an imputation method wins and ties. Then we obtain the average ranking by putting those imputation methods which have a higher "wins + ties" sum first among the rest of the imputation methods. If a draw is found for "wins + ties", we use the "wins" to establish the rank. If some methods obtain a draw for both "wins + ties" and "wins", then an average ranking is assigned for all of them.

In order to compare the imputation methods for the FRBCS considered in each we have added one more final column with the mean ranking for each imputation method across all the data sets, that is, the mean of every row. By doing so, we can obtain a new rank (final column RANKING), in which we propose a new ordering for the imputation methods for a given FRBCS, using the values of the column "Avg." to sort the imputation methods.

Table 5 depicts the results for FH-GBML. The best imputation method is CMC, while DNI is the sixth best. That means that although DNI is capable of obtaining very good results in few data sets, it is a very irregular MV treatment strategy. CMC is capable of obtaining good results in every data set and even being the best on some of them. SVMI and MC imputation methods are also good alternatives for FH-GBML. Case deletion (IM) is an affordable option due to the good generalization abilities of FH-GBML, but it obtains a lower mean accuracy than SVMI and MC.

Table 6 summarize the results for Chi et al. Again we find that CMC is the best imputation choice, while BPCA was the preferred one considering only the accuracy results. The behavior of BPCA is even more irregular for Chi et al. than DNI for FH-GBML. CMC is better than most of the imputation strategies, with 9 wins, making it a clear imputation choice for this FRBCS.

Table 7 shows the results for PDFC. In this case, the ranking using the Wilcoxon statistical comparison is in concordance with the results obtained using the test accuracy: EC is the best imputation method for PDFC with 10 wins out of 14 methods. We can conclude that EC is the best imputation method for PDFC.

For the Wilcoxon tables with their rankings we have built Table 8 with the best three methods of each FRBCS. We have stressed in bold those rankings equal to or below three. An important outcome of the results is that both FH-GBML and Chi et al. FRBCSs share the same best imputation method, while PDFC has a different choice. We must stress that FH-GBML and Chi et al. are Mamdani based FRBCSs, while PDFC is an

Table 5 Average ranks for FH-GBML

	IM	EC	KNNI	WKNNI	KMI	FKMI	SVMI	EM	SVDI	BPCA	LLSI	MC	CMC	DNI	**Ties+Wins**	**Wins**	**RANKING**
IM		D	D	W	D	D	D	W	W	W	D	D	D	D	**13**	**4**	**3**
EC	D		D	D	D	D	D	D	D	W	D	D	D	L	**12**	**1**	**9.5**
KNNI	D	D		D	D	D	D	W	D	W	D	D	D	D	**13**	**2**	**6**
WKNNI	L	D	D		D	D	L	D	D	W	D	L	L	D	**9**	**1**	**11**
KMI	D	D	D	D		D	D	D	D	W	D	D	D	D	**13**	**1**	**8**
FKMI	D	D	D	D	D		D	D	W	W	D	D	D	D	**13**	**2**	**6**
SVMI	D	D	D	W	D	D		W	W	W	D	D	D	D	**13**	**4**	**3**
EM	L	D	L	D	D	D	L		D	W	D	L	L	D	**8**	**1**	**12.5**
SVDI	L	D	D	D	D	L	L	D		W	D	L	L	D	**8**	**1**	**12.5**
BPCA	L	L	L	L	L	L	L	L	L		L	L	L	L	**0**	**0**	**14**
LLSI	D	D	D	D	D	D	D	D	D	W		D	L	D	**12**	**1**	**9.5**
MC	D	D	D	W	D	D	D	W	W	W	D		D	D	**13**	**4**	**3**
CMC	D	D	D	W	D	D	D	W	W	W	W	D		D	**13**	**5**	**1**
DNI	D	W	D	D	D	D	D	D	D	W	D	D	D		**13**	**2**	**6**

Table 7 Average ranks for PDFC

	IM	EC	KNNI	WKNNI	KMI	FKMI	SVMI	EM	SVDI	BPCA	LLSI	MC	CMC	DNI	**Ties+Wins**	**Wins**	**RANKING**
IM		L	D	D	D	D	D	D	D	W	D	D	D	W	**12**	**2**	**7.5**
EC	W		W	W	D	W	D	W	W	W	W	D	W	W	**13**	**10**	**1**
KNNI	D	L		D	D	D	L	D	D	W	D	D	D	W	**11**	**2**	**9.5**
WKNNI	D	L	D		D	D	D	D	D	W	D	D	D	W	**12**	**2**	**7.5**
KMI	D	D	D	D		D	D	D	W	W	D	D	D	W	**13**	**3**	**6**
FKMI	D	L	D	D	D		D	D	W	W	W	D	D	W	**12**	**4**	**4.5**
SVMI	D	D	W	D	D	D		W	W	W	W	D	D	W	**13**	**6**	**2**
EM	D	L	D	D	D	D	L		D	W	D	D	D	W	**11**	**2**	**9.5**
SVDI	D	L	D	D	L	L	L	D		W	D	L	L	W	**7**	**2**	**12**
BPCA	L	L	L	L	L	L	L	L	L		L	L	L	W	**1**	**1**	**13**
LLSI	D	L	D	D	D	L	L	D	D	W		L	L	W	**8**	**2**	**11**
MC	D	D	D	D	D	D	D	D	W	W	W		D	W	**13**	**4**	**3**
CMC	D	L	D	D	D	D	D	D	W	W	W	D		W	**12**	**4**	**4.5**
DNI	L	L	L	L	L	L	L	L	L	L	L	L	L		**0**	**0**	**14**

Table 6 Average ranks for Chi et al.

	IM	EC	KNNI	WKNNI	KMI	FKMI	SVMI	EM	SVDI	BPCA	LLSI	MC	CMC	DNI	**Ties+Wins**	**Wins**	**RANKING**
IM		W	D	D	D	D	D	D	D	D	W	D	D	W	**13**	**3**	**3**
EC	L		L	L	L	L	L	L	L	L	L	L	L	W	**1**	**1**	**13**
KNNI	D	W		D	D	D	L	D	D	D	D	D	L	W	**11**	**2**	**10.5**
WKNNI	D	W	D		D	D	L	D	D	D	D	D	L	W	**11**	**2**	**10.5**
KMI	D	W	D	D		D	D	D	D	D	D	D	L	W	**12**	**2**	**7.5**
FKMI	D	W	D	D	D		D	D	D	D	D	D	L	W	**12**	**2**	**7.5**
SVMI	D	W	W	W	D	D		D	D	D	D	D	D	W	**13**	**4**	**2**
EM	D	W	D	D	D	D	D		W	D	D	D	L	W	**12**	**3**	**4**
SVDI	D	W	D	D	D	D	D	L		D	D	D	L	W	**11**	**2**	**10.5**
BPCA	D	W	D	D	D	D	D	D	D		D	D	D	W	**13**	**2**	**5.5**
LLSI	L	W	D	D	D	D	D	D	D	D		D	L	W	**11**	**2**	**10.5**
MC	D	W	D	D	D	D	D	D	D	D	D		D	W	**13**	**2**	**5.5**
CMC	D	W	W	W	W	W	D	W	W	D	W	D		W	**13**	**9**	**1**
DNI	L	L	L	L	L	L	L	L	L	L	L	L	l		**0**	**0**	**14**

special form of TSK model. Therefore, the kind of FRBCS considered appears to have influence on the best imputation strategy when considering other FRBCSs than those analyzed in this work.

	FH-GBML RANKING	**Chi et al.** RANKING	**PDFC** RANKING
IM	**3**	**3**	**2**
EC	9.5	13	**1**
SVMI	**3**	**2**	**2**
MC	**3**	5.5	**3**
CMC	**1**	**1**	4.5

Table 8 Best imputation methods for FRBCS

As a final remark, we can state that:

- The imputation methods which fill in the MVs outperform the case deletion (IM method) and the lack of imputation (DNI method). Only in the case of the IM imputation method obtain a relatively low rank (2nd and 3th place) but these results can be altered when new examples are presented to the model learned with less data. This fact indicates that the imputation methods usually outperform the non-imputation strategies.
- There is no universal imputation method which performs best for all type of FRBCS.

Please note that we have tackled the second point by adding a categorization and a wide benchmark bed, obtaining a group of recommended imputation methods for each family.

6 Influence of the imputation on the instances and individual features

In the previous section we have analyzed the relationship between the use of several imputation methods with respect to the FRBCS's accuracy. However, it would be interesting to relate the influence of the imputation methods to the information contained in the data set. In order to study the influence and the benefits/drawbacks of using the different imputation methods, we have considered the use of two different measures. They are described as follows:

- Wilson's Noise Ratio: This measure proposed by Wilson (1972) observes the noise in the data set. For each instance of interest, the method looks for the K nearest neighbors (using the euclidean distance), and uses the class labels of such neighbors in order to classify the considered instance. If the instance is not correctly classified, then the variable *noise* is increased by one unit. Therefore, the final noise ratio will be

$$\text{Wilson's Noise} = \frac{noise}{\#\text{ instances in the data set}}$$

  In particular, we only compute the noise for the *imputed* instances considering $K = 5$.
- Mutual Information: Mutual information (MI) is considered to be a good indicator of relevance between two random variables (Cover and Thomas 1991). Recently, the use of the MI measure in feature selection has become well-known and seen to be successful (Kwak and Choi 2002b,a; Peng et al 2005). The use of the MI measure for continuous attributes has been tackled by (Kwak and Choi 2002a), allowing us to compute the MI measure not only in nominal-valued data sets.
  In our approach, we calculate the MI between each input attribute and the class attribute, obtaining a set of values, one for each input attribute. In the next step we compute the ratio between each one of these values, considering the imputation of the data set with one imputation method in respect to the not imputed data set. The average of these ratios will show us if the imputation of the data set produces a gain in information:

$$\text{Avg. MI Ratio} = \frac{\sum_{x_i \in X} \frac{MI_\alpha(x_i)+1}{MI(x_i)+1}}{|X|}$$

  where X is the set of input attributes, $MI_\alpha(i)$ represents the MI value of the ith attribute in the imputed data set and $MI(i)$ is the MI value of the ith input attribute in the not imputed data set. We have also applied the Laplace correction, summing 1 to both numerator and denominator, as an MI value of zero is possible for some input attributes.
  The calculation of $MI(x_i)$ depends on the type of attribute x_i. If the attribute x_i is nominal, the MI between x_i and the class label Y is computed as follows:

$$MI_{nominal}(x_i) = I(x_i;Y) = \sum_{z \in x_i} \sum_{y \in Y} p(z,y) log_2 \frac{p(z,y)}{p(z)p(y)}.$$

  On the other hand, if the attribute x_i is numeric, we have used the Parzen window density estimate as shown in (Kwak and Choi 2002a) considering a Gaussian window function:

$$MI_{numeric}(x_i) = I(x_i;Y) = H(Y) - H(C|X);$$

  where $H(Y)$ is the entropy of the class label

$$H(Y) = -\sum_{y \in Y} p(y) log_2 p(y);$$

Table 9 Wilson's noise ratio values

Data-set	Imp. Method	% Wilson's Noise Ratio	Data-set	Imp. Method	% Wilson's Noise Ratio	Data-set	Imp. Method	% Wilson's Noise Ratio
CLE	MC	50.0000	HOV	MC	7.9208	HEP	MC	17.3333
	CMC	50.0000		CMC	5.4455		CMC	**16.0000**
	KNNI	50.0000		KNNI	7.4257		KNNI	20.0000
	WKNNI	50.0000		WKNNI	7.4257		WKNNI	20.0000
	KMI	50.0000		KMI	7.4257		KMI	20.0000
	FKMI	50.0000		FKMI	7.9208		FKMI	17.3333
	SVMI	50.0000		SVMI	6.9307		SVMI	17.3333
	EM	66.6667		EM	11.8812		EM	22.6667
	SVDI	66.6667		SVDI	8.9109		SVDI	21.3333
	BPCA	50.0000		BPCA	6.9307		BPCA	21.3333
	LLSI	50.0000		LLSI	**4.9505**		LLSI	18.6667
	EC	**33.3333**		EC	7.4257		EC	**16.0000**
WIS	MC	18.7500	WAT	MC	31.5068	MUS	MC	**0.0000**
	CMC	**12.5000**		CMC	**21.2329**		CMC	**0.0000**
	KNNI	**12.5000**		KNNI	27.3973		KNNI	**0.0000**
	WKNNI	**12.5000**		WKNNI	27.3973		WKNNI	**0.0000**
	KMI	**12.5000**		KMI	27.3973		KMI	**0.0000**
	FKMI	**12.5000**		FKMI	31.5068		FKMI	**0.0000**
	SVMI	**12.5000**		SVMI	23.9726		SVMI	**0.0000**
	EM	**12.5000**		EM	46.5753		EM	**0.0000**
	SVDI	**12.5000**		SVDI	49.3151		SVDI	**0.0000**
	BPCA	**12.5000**		BPCA	26.0274		BPCA	**0.0000**
	LLSI	**12.5000**		LLSI	25.3425		LLSI	**0.0000**
	EC	**12.5000**		EC	22.6027		EC	**0.0000**
CRX	MC	18.9189	SPO	MC	27.2727	POS	MC	**33.3333**
	CMC	18.9189		CMC	**22.7273**		CMC	**33.3333**
	KNNI	21.6216		KNNI	27.2727		KNNI	**33.3333**
	WKNNI	21.6216		WKNNI	27.2727		WKNNI	**33.3333**
	KMI	21.6216		KMI	27.2727		KMI	**33.3333**
	FKMI	18.9189		FKMI	27.2727		FKMI	**33.3333**
	SVMI	13.5135		SVMI	27.2727		SVMI	**33.3333**
	EM	32.4324		EM	36.3636		EM	**33.3333**
	SVDI	27.0270		SVDI	31.8182		SVDI	**33.3333**
	BPCA	21.6216		BPCA	27.2727		BPCA	**33.3333**
	LLSI	18.9189		LLSI	27.2727		LLSI	**33.3333**
	EC	**13.5135**		EC	27.2727		EC	**33.3333**
BRE	MC	55.5556	BAN	MC	25.4753	ECH	MC	40.0000
	CMC	55.5556		CMC	24.3346		CMC	40.0000
	KNNI	55.5556		KNNI	23.1939		KNNI	46.6667
	WKNNI	55.5556		WKNNI	22.8137		WKNNI	44.4444
	KMI	55.5556		KMI	25.4753		KMI	46.6667
	FKMI	55.5556		FKMI	24.3346		FKMI	40.0000
	SVMI	55.5556		SVMI	**21.2928**		SVMI	44.4444
	EM	**44.4444**		EM	26.2357		EM	51.1111
	SVDI	**44.4444**		SVDI	22.4335		SVDI	48.8889
	BPCA	66.6667		BPCA	23.9544		BPCA	44.4444
	LLSI	66.6667		LLSI	24.7148		LLSI	**37.7778**
	EC	66.6667		EC	23.5741		EC	48.8889
AUT	MC	45.6522	HOC	MC	19.3906	SOY	MC	**2.4390**
	CMC	41.3043		CMC	**10.2493**		CMC	**2.4390**
	KNNI	41.3043		KNNI	20.2216		KNNI	**2.4390**
	WKNNI	41.3043		WKNNI	19.1136		WKNNI	**2.4390**
	KMI	41.3043		KMI	21.8837		KMI	**2.4390**
	FKMI	45.6522		FKMI	20.4986		FKMI	**2.4390**
	SVMI	43.4783		SVMI	20.2216		SVMI	**2.4390**
	EM	58.6957		EM	21.0526		EM	**2.4390**
	SVDI	52.1739		SVDI	21.0526		SVDI	7.3171
	BPCA	43.4783		BPCA	19.3906		BPCA	7.3171
	LLSI	45.6522		LLSI	20.4986		LLSI	**2.4390**
	EC	**30.4348**		EC	20.7756		EC	**2.4390**
PRT	MC	71.0145	AUD	MC	38.7387	MAM	MC	21.3740
	CMC	**60.8696**		CMC	**32.8829**		CMC	**13.7405**
	KNNI	69.5652		KNNI	38.7387		KNNI	25.9542
	WKNNI	69.5652		WKNNI	38.7387		WKNNI	25.9542
	KMI	71.0145		KMI	38.7387		KMI	24.4275
	FKMI	71.0145		FKMI	38.7387		FKMI	20.6107
	SVMI	68.1159		SVMI	37.8378		SVMI	16.7939
	EM	88.4058		EM	53.6036		EM	20.6107
	SVDI	91.7874		SVDI	46.3964		SVDI	27.4809
	BPCA	71.4976		BPCA	40.5405		BPCA	25.1908
	LLSI	69.5652		LLSI	36.9369		LLSI	26.7176
	EC	66.1836		EC	37.8378		EC	18.3206
DER	MC	**0.0000**	LUN	MC	80.0000	OZO	MC	4.8035
	CMC	**0.0000**		CMC	80.0000		CMC	**3.6390**
	KNNI	**0.0000**		KNNI	80.0000		KNNI	4.3668
	WKNNI	**0.0000**		WKNNI	80.0000		WKNNI	4.5124
	KMI	**0.0000**		KMI	80.0000		KMI	4.9491
	FKMI	**0.0000**		FKMI	80.0000		FKMI	4.0757
	SVMI	**0.0000**		SVMI	80.0000		SVMI	3.7846
	EM	**0.0000**		EM	**20.0000**		EM	4.8035
	SVDI	**0.0000**		SVDI	40.0000		SVDI	4.8035
	BPCA	**0.0000**		BPCA	80.0000		BPCA	4.3668
	LLSI	**0.0000**		LLSI	80.0000		LLSI	4.2213
	EC	**0.0000**		EC	80.0000		EC	4.8035

and $H(C|X)$ is the conditional entropy

$$H(Y|x_i) = -\sum_{z \in x_i} \sum_{y \in Y} p(z,y) log_2 p(y|z).$$

Considering that each sample has the same probability, applying the Bayesian rule and approximating $p(y|z)$ by the Parzen window we get:

$$\hat{H}(Y|x_i) = -\sum_{j=1}^{n} \frac{1}{n} \sum_{y=1}^{N} \hat{p}(y|z_j) log_2 \hat{p}(y|z_j)$$

where n is the number of instances in the data set, N is the total number of class labels and $\hat{p}(c|x)$ is

Table 10 Average mutual information ratio

Data-set	Imp. Method	Avg. MI ratio	Data-set	Imp. Method	Avg. MI ratio	Data-set	Imp. Method	Avg. MI ratio
CLE	MC	0.998195	HOV	MC	0.961834	HEP	MC	0.963765
	CMC	0.998585		CMC	1.105778		CMC	0.990694
	KNNI	0.998755		KNNI	0.965069		KNNI	0.978564
	WKNNI	0.998795		WKNNI	0.965069		WKNNI	0.978343
	KMI	0.998798		KMI	0.961525		KMI	0.980094
	FKMI	0.998889		FKMI	0.961834		FKMI	0.963476
	SVMI	0.998365		SVMI	0.908067		SVMI	1.006819
	EM	0.998152		EM	0.891668		EM	0.974433
	SVDI	0.997152		SVDI	0.850361		SVDI	0.967673
	BPCA	0.998701		BPCA	1.091675		BPCA	0.994420
	LLSI	0.998882		LLSI	**1.122904**		LLSI	0.995464
	EC	**1.000148**		EC	1.007843		EC	**1.024019**
WIS	MC	0.999004	WAT	MC	0.959488	MUS	MC	1.018382
	CMC	0.999861		CMC	0.967967		CMC	1.018382
	KNNI	0.999205		KNNI	0.961601		KNNI	0.981261
	WKNNI	0.999205		WKNNI	0.961574		WKNNI	0.981261
	KMI	0.999322		KMI	0.961361		KMI	1.018382
	FKMI	0.998923		FKMI	0.961590		FKMI	1.018382
	SVMI	0.999412		SVMI	0.967356		SVMI	0.981261
	EM	0.990030		EM	0.933846		EM	**1.142177**
	SVDI	0.987066		SVDI	0.933040		SVDI	1.137152
	BPCA	0.998951		BPCA	0.964255		BPCA	0.987472
	LLSI	0.999580		LLSI	0.964063		LLSI	0.977275
	EC	**1.000030**		EC	**1.027369**		EC	1.017366
CRX	MC	1.000883	SPO	MC	0.997675	POS	MC	1.012293
	CMC	1.000966		CMC	**1.022247**		CMC	1.012293
	KNNI	0.998823		KNNI	0.999041		KNNI	1.012293
	WKNNI	0.998870		WKNNI	0.999041		WKNNI	1.012293
	KMI	1.001760		KMI	0.998464		KMI	1.012293
	FKMI	1.000637		FKMI	0.997675		FKMI	1.012293
	SVMI	0.981878		SVMI	1.015835		SVMI	1.012293
	EM	0.985609		EM	0.982325		EM	1.012293
	SVDI	0.976398		SVDI	0.979187		SVDI	1.014698
	BPCA	0.999934		BPCA	1.006236		BPCA	1.012293
	LLSI	1.001594		LLSI	1.004821		LLSI	**1.018007**
	EC	**1.008718**		EC	1.018620		EC	0.997034
BRE	MC	0.998709	BAN	MC	1.012922	ECH	MC	0.981673
	CMC	0.998709		CMC	1.070857		CMC	0.995886
	KNNI	0.992184		KNNI	0.940369		KNNI	0.997912
	WKNNI	0.992184		WKNNI	0.940469		WKNNI	**0.998134**
	KMI	0.998709		KMI	1.016101		KMI	0.967169
	FKMI	0.998709		FKMI	1.020989		FKMI	0.983606
	SVMI	0.998709		SVMI	**1.542536**		SVMI	0.987678
	EM	**1.013758**		EM	1.350315		EM	0.967861
	SVDI	0.999089		SVDI	1.365572		SVDI	0.935855
	BPCA	1.000201		BPCA	1.010596		BPCA	0.972327
	LLSI	1.000201		LLSI	1.015033		LLSI	0.988591
	EC	1.001143		EC	1.102328		EC	0.970029
AUT	MC	0.985610	HOC	MC	0.848649	SOY	MC	1.056652
	CMC	0.991113		CMC	**2.039992**		CMC	1.123636
	KNNI	0.986239		KNNI	0.834734		KNNI	1.115818
	WKNNI	0.985953		WKNNI	0.833982		WKNNI	1.115818
	KMI	0.985602		KMI	0.821936		KMI	1.056652
	FKMI	0.984694		FKMI	0.849141		FKMI	1.056652
	SVMI	0.991850		SVMI	0.843456		SVMI	**1.772589**
	EM	0.970557		EM	0.775773		EM	1.099286
	SVDI	0.968938		SVDI	0.750930		SVDI	1.065865
	BPCA	0.986631		BPCA	0.964587		BPCA	1.121603
	LLSI	0.985362		LLSI	0.926068		LLSI	1.159610
	EC	**1.007652**		EC	0.911543		EC	1.222631
PRT	MC	0.949896	AUD	MC	0.990711	MAM	MC	0.974436
	CMC	**1.120006**		CMC	1.032162		CMC	1.029154
	KNNI	0.976351		KNNI	0.993246		KNNI	0.965926
	WKNNI	0.976351		WKNNI	0.993246		WKNNI	0.965926
	KMI	0.949896		KMI	1.000235		KMI	0.966885
	FKMI	0.949896		FKMI	0.990711		FKMI	0.974228
	SVMI	1.038152		SVMI	1.007958		SVMI	**1.272993**
	EM	0.461600		EM	1.129168		EM	0.980865
	SVDI	0.485682		SVDI	1.065091		SVDI	1.052790
	BPCA	0.987598		BPCA	1.156676		BPCA	0.978209
	LLSI	1.016230		LLSI	1.061197		LLSI	0.994349
	EC	1.053185		EC	**1.209608**		EC	1.269505
DER	MC	1.000581	LUN	MC	0.996176	OZO	MC	0.982873
	CMC	**1.002406**		CMC	1.008333		CMC	**0.989156**
	KNNI	0.999734		KNNI	0.996176		KNNI	0.982759
	WKNNI	0.999734		WKNNI	0.996176		WKNNI	0.982721
	KMI	1.000581		KMI	0.996176		KMI	0.982495
	FKMI	1.000581		FKMI	0.996176		FKMI	0.982951
	SVMI	1.001566		SVMI	1.006028		SVMI	0.988297
	EM	1.000016		EM	1.067844		EM	0.979977
	SVDI	0.999691		SVDI	**1.076334**		SVDI	0.979958
	BPCA	0.999633		BPCA	0.996447		BPCA	0.983318
	LLSI	0.999170		LLSI	1.007612		LLSI	0.983508
	EC	1.000539		EC	1.002385		EC	0.944747

$$\hat{p}(y|z) = \frac{\sum_{i \in I_c} exp\left(-\frac{(z-z_i)\Sigma^{-1}(z-z_i)}{2h^2}\right)}{\sum_{k=1}^{N}\sum_{i \in I_k} exp\left(-\frac{(z-z_i)\Sigma^{-1}(z-z_i)}{2h^2}\right)}.$$

In this case, I_c is the set of indices of the training examples belonging to class c, and Σ is the covariance of the random variable $(z - z_i)$.

Comparing with Wilson's noise ratio we can observe which imputation methods reduce the impact of the MVs as a noise, and which methods produce noise when imputing. In addition the MI ratio allows us to relate the attributes to the imputation results. A value of the MI ratio higher than 1 will indicate that the imputation is capable of relating more of the attributes individually to the class labels. A value lower than 1 will indicate

that the imputation method is adversely affecting the relationship between the individual attributes and the class label.

In Table 9 we have summarized the Wilson's noise ratio values for the 21 data sets considered in our study. We must point out that the results of Wilson's noise ratio are related to a given data set. Hence, the characteristics of the proper data appear to determine the values of this measure.

In Table 10 we have summarized the average MI ratios for the 21 data sets. In the results we can observe that the average ratios are usually close to 1; that is, the use of imputation methods appears to harm the relationship between the class label and the input attribute little or not at all, even improving it in some cases. However, the mutual information considers only one attribute at a time and therefore the relationships between the input attributes are ignored. The imputation methods estimate the MVs using such relationships and can afford improvements in the performance of the FRBCSs. Hence the highest values of average MI ratios could be related to those methods which can obtain better estimates for the MVs, and maintaining the relationship degree between the class labels and the isolated input attributes. It is interesting to note that when analyzing the MI ratio, the values do not appear to be as highly data dependant as Wilson's noise ratio, as the values for all the data sets are more or less close to each other.

If we count the methods with the lowest Wilson's noise ratios in each data set in Table 9, we find that the CMC method is first, with 12 times the lowest one, and the EC method is second with 9 times the lowest one. If we count the methods with the highest mutual information ratio in each data set in Table 10, the EC method has the highest ratio for 7 data sets and is therefore the first one. The CMC method has the highest ratio for 5 data sets and is the second one in this case. Considering the analysis of the previous Subsection 5.2 with these two methods:

- The EC method is the best method obtained for PDFC, and the third best for the Rule Induction Learning methods while is one of the worst for Chi et al. and PDFC methods. Therefore the TSK models seems to benefit more from those imputation methods which produce gain in the MI.
- The CMC method is the best method for the Mamdani models (Chi et al. and FH-GBML), and not very bad for PDFC. Mamdani FRBCSs benefit from the imputation method which induce less noise in the resultant imputed data set.

Next, we rank all the imputation methods according to the values presented in Tables 9 and 10. In order to do so, we have calculated the average rankings of each imputation method for all the data sets, for both Wilson's noise ratio and the mutual information ratio. The method to compute this average ranking is the same as that presented in Subsection 5.2. In Table 11 we have gathered together these average rankings, as well as their relative position in parentheses.

Table 11 Average rankings for Wilson's noise ratio and Mutual information ratio

	Avg. Rankings	
	Wilson's noise ratio	Mutual information
MC	6.98 (8)	8.05 (11)
CMC	**3.79** (1)	**3.60** (1)
KNNI	6.43 (7)	7.69 (8)
WKNNI	6.17 (5)	7.79 (9)
KMI	7.38 (10)	7.60 (6)
FKMI	6.36 (6)	7.62 (7)
SVMI	4.67 (2)	4.90 (4)
EM	8.93 (12)	7.90 (10)
SVDI	8.86 (11)	8.48 (12)
BPCA	7.17 (9)	5.79 (5)
LLSI	5.98 (4)	4.74 (3)
EC	**5.31** (3)	**3.86** (2)

From the average rankings shown in Table 11, we can observe that the CMC method is the first for both rankings. The EC method is the second for the mutual information ratio, and the third one for Wilson's noise ratio. The SVMI method obtains the second lowest ranking for Wilson's noise ratio, and the fourth lowest ranking for the MI ratio. The SVMI method is the second best method for the Rule Induction Learning algorithms with average rankings close to EC.

With the analysis performed we have quantified the noise induced by the imputation methods and how the relationship between each input attribute and the class is maintained. We have discovered that the CMC and EC methods show good behavior for these two measures, and they are two methods that the best results for the FRBCSs as we have previously analyzed. In short, these two approaches introduce less noise and maintain the mutual information better. They can provide us with a first characterization of imputation methods and a first step for providing us with tools for analyzing the imputation method's behavior.

7 Concluding remarks

This study is a general comparison of FRBCSs not previously considered in MV studies. We have studied the use of imputation techniques for the analysis of three representative FRBCSs, presenting an analysis among imputation, do not impute and ignore cases with MVs.

We have used a large bunch of data sets with real MVs to do so.

From the obtained results in Section 5.2, the particular analysis of the MVs treatment methods conditioned to the FRBCS nature is necessary. Thus, we can stress particular imputation algorithms based on the classification groups, as in the case of the *CMC* method for the Mamdami FRBCSs and the *EC* method for the TSK models. Therefore, we can confirm the positive effect of the imputation methods and the FRBCS' behavior, and the presence of more suitable imputation methods for some particular FRBCS categories than others.

Moreover, we have analyzed the influence of the imputation methods in respect to two measures. These two measures are the *Wilson's noise ratio* and the *average mutual information difference*. The first one quantifies the noise induced by the imputation method in the instances which contain MVs. The second one examines the increment or decrement in the relationship of the isolated input attributes with respect to the class label. We have observed that the CMC and EC methods are the ones which introduce less noise and maintain the mutual information better, which correspond to the best imputation methods observed for each FRBCS types.

Acknowledgements

This work was supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01. J. Luengo and J.A. Sáez hold a FPU scholarship from Spanish Ministry of Education and Science.

References

Acuna E, Rodriguez C (2004) The treatment of missing values and its effect in the classifier accuracy. In: Banks D, House L, McMorris F, Arabie P, Gaul W (eds) Classification, Clustering and Data Mining Applications, Springer-Verlag Berlin-Heidelberg, pp 639–648

Alcalá-Fdez J, Sánchez L, García S, Jesus MJD, Ventura S, Garrell JM, Otero J, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) Keel: A software tool to assess evolutionary algorithms for data mining problems. Soft Computing 13(3):307–318

Barnard J, Meng X (1999) Applications of multiple imputation in medical studies: From AIDS to NHANES. Statistical Methods in Medical Research 8(1):17–36

Batista G, Monard M (2003) An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence 17(5):519–533

Berthold MR, Huber KP (1998) Missing values and learning of fuzzy rules. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6:171–178

Chen Y, Wang JZ (2003) Support vector learning for fuzzy rule-based classification systems. IEEE Transactions on Fuzzy Systems 11(6):716–728

Chi Z, Yan H, Pham T (1996) Fuzzy algorithms with applications to image processing and pattern recognition. World Scientific

Cover TM, Thomas JA (1991) Elements of Information Theory, 2nd edn. John Wiley

Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, New York, NY, USA

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7:1–30

Dubois D, Prade H (1978) Operations on fuzzy numbers. International Journal of Systems Sciences 9:613–626

Farhangfar A, Kurgan LA, Pedrycz W (2007) A novel framework for imputation of missing values in databases. IEEE Transactions on Systems, Man, and Cybernetics, Part A 37(5):692–709

Farhangfar A, Kurgan L, Dy J (2008) Impact of imputation of missing values on classification error for discrete data. Pattern Recognition 41(12):3692–3705

Feng H, Guoshun C, Cheng Y, Yang B, Chen Y (2005) A SVM regression based approach to filling in missing values. In: Khosla R, Howlett RJ, Jain LC (eds) 9th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES 2005), Springer, Lecture Notes in Computer Science, vol 3683, pp 581–587

Gabriel TR, Berthold MR (2005) Missing values in fuzzy rule induction. In: Anderson G, Tunstel E (eds) 2005 IEEE Conference on Systems, Man and Cybernetics, IEEE Press

García S, Herrera F (2008) An extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all pairwise comparisons. Journal of Machine Learning Research 9:2677–2694

García-Laencina P, Sancho-Gómez J, Figueiras-Vidal A (2009) Pattern classification with missing data: a review. Neural Computation & Applications 9(1):1–12

Gheyas IA, Smith LS (2010) A neural network-based framework for the reconstruction of incomplete data sets. Neurocomputing 73(16-18):3039–3065

Grzymala-Busse J, Goodwin L, Grzymala-Busse W, Zheng X (2005) Handling missing attribute values in preterm birth data sets. In: 10th International Conference of Rough Sets and Fuzzy Sets and Data Mining and Granular Computing(RSFDGrC'05), pp 342–351

Grzymala-Busse JW, Hu M (2000) A comparison of several approaches to missing attribute values in data mining. In: Ziarko W, Yao YY (eds) Rough Sets and Current Trends in Computing, Springer, Lecture Notes in Computer Science, vol 2005, pp 378–385

Ishibuchi H, Nakashima T, Nii M (2004) Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining. Springer-Verlag New York, Inc.

Ishibuchi H, Yamamoto T, Nakashima T (2005) Hybridization of fuzzy GBML approaches for pattern classification problems. IEEE Transactions on System, Man and Cybernetics B 35(2):359–365

Jr ERH, Hruschka ER, Ebecken NFF (2007) Bayesian networks for imputation in classification problems. Journal of Intelligent Information Systems 29(3):231–252

Kim H, Golub GH, Park H (2005) Missing value estimation for DNA microarray gene expression data: local least squares imputation. Bioinformatics 21(2):187–198

Kuncheva L (2000) Fuzzy Classifier Design. Springer, Berlin

Kwak N, Choi CH (2002a) Input feature selection by mutual information based on parzen window. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(12):1667–1671

Kwak N, Choi CH (2002b) Input feature selection for classification problems. IEEE Transactions on Neural Networks 13(1):143–159

Li D, Deogun J, Spaulding W, Shuart B (2004) Towards missing data imputation: A study of fuzzy k-means clustering method. In: 4th International Conference of Rough Sets and Current Trends in Computing(RSCTĆ04), pp 573–579

Little RJA, Rubin DB (1987) Statistical Analysis with Missing Data, 1st edn. Wiley Series in Probability and Statistics, Wiley, New York

Luengo J, García S, Herrera F (2010) A study on the use of imputation methods for experimentation with radial basis function network classifiers handling missing attribute values: The good synergy between RBFNs and EventCovering method. Neural Networks 23:406–418

Matsubara ET, Prati RC, Batista GEAPA, Monard MC (2008) Missing value imputation using a semi-supervised rank aggregation approach. In: Zaverucha G, da Costa ACPL (eds) 19th Brazilian Symposium on Artificial Intelligence (SBIA 2008), Springer, Lecture Notes in Computer Science, vol 5249, pp 217–226

Oba S, aki Sato M, Takemasa I, Monden M, ichi Matsubara K, Ishii S (2003) A bayesian missing value estimation method for gene expression profile data. Bioinformatics 19(16):2088–2096

Peng H, Long F, Ding C (2005) Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(8):1226–1238

Platt JC (1999) Fast training of support vector machines using sequential minimal optimization. In: Advances in kernel methods: support vector learning, MIT Press, Cambridge, MA, USA, pp 185–208

Pyle D (1999) Data Preparation for Data Mining. Morgan Kaufmann

Schafer JL (1997) Analysis of Incomplete Multivariate Data. Chapman & Hall, London

Schneider T (2001) Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. Journal of Climate 14:853–871

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB (2001) Missing value estimation methods for dna microarrays. Bioinformatics 17(6):520–525

Vapnik VN (1998) Statistical Learning Theory. Wiley-Interscience

Wang H, Wang S (2010) Mining incomplete survey data through classification. Knowledge and Information Systems 24(2):221–233

Wang LX, Mendel JM (1992) Generating fuzzy rules by learning from examples. IEEE Transactions on Systems, Man, and Cybernetics 25(2):353–361

Wilson D (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems and Man and Cybernetics 2(3):408–421

Wong AKC, Chiu DKY (1987) Synthesizing statistical knowledge from incomplete mixed-mode data. IEEE Transactions on Pattern Analysis and Machine Intelligence 9(6):796–805

2. Domains of Competence of Fuzzy Rule Based Classification Systems: An Ad-hoc and an Automatic Approach

The journal papers associated to this part are:

2.1. Domains of Competence of Fuzzy Rule Based Classification Systems with Data Complexity measures: A case of study using a Fuzzy Hybrid Genetic Based Machine Learning Method

- J. Luengo, F. Herrera, Domains of Competence of Fuzzy Rule Based Classification Systems with Data Complexity measures: A case of study using a Fuzzy Hybrid Genetic Based Machine Learning Method. Fuzzy Sets and Systems, 161 (1) (2010) 3-19 doi:10.1016/j.fss.2009.04.001.

  - Status: **Published**.
  - Impact Factor (JCR 2009): 2.138.
  - Subject Category: Computer Science, Theory & Methods. Ranking 14 / 92.
  - Subject Category: Mathematics, Applied. Ranking 8 / 204.
  - Subject Category: Statistics & Probability. Ranking 13 / 100.

Available online at www.sciencedirect.com
ScienceDirect
Fuzzy Sets and Systems 161 (2010) 3 – 19
FUZZY sets and systems
www.elsevier.com/locate/fss

Domains of competence of fuzzy rule based classification systems with data complexity measures: A case of study using a fuzzy hybrid genetic based machine learning method☆

Julián Luengo*, Francisco Herrera

Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

Available online 15 April 2009

Abstract

The analysis of data complexity is a proper framework to characterize the tackled classification problem and to identify domains of competence of classifiers. As a practical outcome of this framework, the proposed data complexity measures may facilitate the choice of a classifier for a given problem. The aim of this paper is to study the behaviour of a fuzzy rule based classification system and its relationship to data complexity. We use as a case of study the fuzzy hybrid genetic based machine learning method presented in [H. Ishibuchi, T. Yamamoto, T. Nakashima, Hybridization of fuzzy GBML approaches for pattern classification problems, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 35 (2) (2005) 359–365]. We examine several metrics of data complexity over a wide range of data sets built from real data and try to extract behaviour patterns from the results. We obtain rules which describe both good or bad behaviours of the fuzzy rule based classification system. These rules use values of data complexity metrics in their antecedents, so we try to predict the behaviour of the method from the data set complexity metrics prior to its application. Therefore, we can establish the domains of competence of this fuzzy rule based classification system.

Keywords: Classification; Data complexity; Fuzzy rule based systems; Genetic fuzzy systems

1. Introduction

Fuzzy rule based classification systems (FRBCSs) [17,19] are a very useful tool in the ambit of machine learning since they are capable of building a linguistic model clearly interpretable by human beings. There is a vast literature in the field of FRBCSs [19], which is very active at this time.

The prediction capabilities of classifiers are strongly dependent on the problem's characteristics. An emergent field has recently arisen, that uses a set of complexity measures applied to the problem to describe its difficulty. These measures quantify particular aspects of the problem which are considered complicated to the classification task [15]. Studies of data complexity metrics applied to particular classification algorithms can be found in [15,3,2,22].

☆ Supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01. J. Luengo holds an FPU scholarship from Spanish Ministry of Education and Science.
* Corresponding author. Tel.: +34 958240598.
E-mail addresses: julianlm@decsai.ugr.es (J. Luengo), herrera@decsai.ugr.es (F. Herrera).

0165-0114/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2009.04.001

The complexity in the data can be used for characterizing FRBCS performance and it can be considered a new trend in the use of FRBCSs in pattern recognition. We understand that no data complexity metrics have been analysed with FRBCSs up to now.

In this work we are interested in analysing the relationship between FRBCSs and the complexity measures, considering a case of study using the fuzzy hybrid genetic based machine learning (FH-GBML) method proposed by Ishibuchi and Yamamoto [18]. In particular we consider three types of data complexity measures based on the overlaps in feature values from different classes; separability of classes; and measures of geometry, topology, and density of manifolds.

To perform this study, we have created 438 binary classification data sets from real world problems, and computed the value of eight metrics proposed by Ho and Basu [14]. We have analysed the intervals of the complexity measure values related to the created data sets, in which FH-GBML method performs well or badly, and then formulated a rule for such intervals. The rules try to describe the ranges where some information and conclusions about the behaviour of the FH-GBML method can be stated.

The paper is organized as follows. In Section 2 we describe the FRBCS we have used. In Section 3 the considered complexity measures are introduced as well as the most recent literature on the topic. In Section 4 we show the process used to build up the bunch of data sets used and the validation scheme. In Section 5 we include the experimental set-up and the results obtained and rules extracted, along with their analysis. Finally, in Section 6 some concluding remarks are pointed out.

2. Preliminaries: fuzzy rule based classification systems

Any classification problem consists of m training patterns $x_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, m$ from M classes where x_{pi} is the ith attribute value ($i = 1, 2, \ldots, n$) of the p-th training pattern.

In this work we use fuzzy rules of the following form:

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then Class} = C_j \text{ with } RW_j, \tag{1}$$

where R_j is the label of the jth rule, $x = (x_1, \ldots, x_n)$ is an n-dimensional pattern vector, A_{ji} is an antecedent fuzzy set, C_j is a class label, and RW_j is the rule weight. We use triangular membership functions as antecedent fuzzy sets.

As learning method we use FH-GBML [18], which belongs to the genetic fuzzy system (GFS) [8]. In the following three subsections we include the fuzzy reasoning model, a complete description of the algorithm, and a short review on the GFSs topic.

2.1. Fuzzy reasoning model

Considering a new pattern $x_p = (x_{p1}, \ldots, x_{pn})$ and a rule base (RB) composed of L fuzzy rules, the steps followed by the reasoning model are the following [7]:

1. *Matching degree*. To calculate the *strength of activation of the if-part for all rules in the RB with the pattern* x_p, using a conjunction operator (usually a T-norm):

$$\mu_{A_j}(x_p) = T(\mu_{A_{j1}}(x_{p1}), \ldots, \mu_{A_{jn}}(x_{pn})), \quad j = 1, \ldots, L. \tag{2}$$

   In this work we will use the product T-norm.
2. *Association degree*. To compute the *association degree of the pattern* x_p *with the M classes according to each rule in the RB*. When using rules with the form of (1), this association degree only refers to the consequent class of the rule (i.e., $k = C_j$):

$$b_j^k = h(\mu_{A_j}(x_p),\ RW_j^k), \quad k = 1, \ldots, M,\ \ j = 1, \ldots, L. \tag{3}$$

   We model function h as the product T-norm in every case.
3. *Pattern classification soundness degree for all classes*. We use an aggregation function that combines the positive degrees of association calculated in the previous step:

$$Y_k = f(b_j^k,\ j = 1, \ldots, L \text{ and } b_j^k > 0), \quad k = 1, \ldots, M. \tag{4}$$

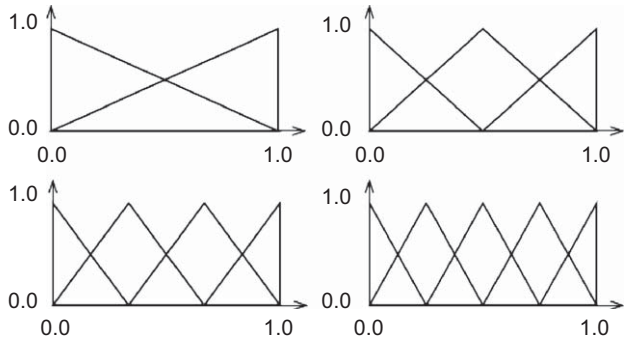

Fig. 1. Four fuzzy partitions for each attribute membership function.

As fuzzy reasoning method we use the winner rule method (classical approach) for classifying new patterns with the rule set. Every new pattern is classified as the consequent class of a single winner rule which is determined as

$$Y_k = \max\{b_j^k, \quad j = 1, \ldots, L \text{ and } k = C_j\}. \tag{5}$$

4. *Classification*. We apply a decision function F over the soundness degree of the system for the pattern classification for all classes. This function will determine the class label l corresponding to the maximum value:

$$F(Y_1, \ldots, Y_M) = l \quad \text{such that } Y_l = \{\max(Y_k), k = 1, \ldots, M\}. \tag{6}$$

2.2. Learning approach: fuzzy hybrid genetic based machine learning method

The basis of this algorithm described here, FH-GBML, consists of a hybrid Pittsburgh and Michigan genetic learning approach [18]:

- The Pittsburgh approach in which each rule set is handled as an individual.
- The Michigan approach (where an individual represents an unique rule), which is used as a kind of heuristic mutation for partially modifying each rule set, because of its high search ability to efficiently find good fuzzy rules.

This method simultaneously uses four fuzzy set partitions for each attribute, as shown in Fig. 1. As a result, each antecedent attribute is initially associated with 14 fuzzy sets generated by these four partitions as well as a special "do not care" set (i.e., 15 in total).

The main steps of this algorithm are described below:

Step 1: Generate N_{pop} rule sets with N_{rule} fuzzy rules.

Step 2: Calculate the fitness value of each rule set in the current population.

Step 3: Generate $(N_{pop} - 1)$ rule sets by the selection, crossover and mutation in the same manner as the Pittsburgh-style algorithm. Apply a single iteration of the Michigan-style algorithm (i.e., the rule generation and the replacement) to each of the generated rule sets with a pre-specified probability.

Step 4: Add the best rule set in the current population to the newly generated $(N_{pop} - 1)$ rule sets to form the next population.

Step 5: Return to Step 2 if the pre-specified stopping condition is not satisfied.

Next, we will describe every step of the algorithm:

- Initialization: N_{rule} training patterns are randomly selected. Then, a fuzzy rule from each of the selected training patterns is generated by choosing probabilistically (as shown in (7)) an antecedent fuzzy set from the 14 candidates $B_k(k = 1, 2, \ldots, 14)$ (see Fig. 1) for each attribute. Then each antecedent fuzzy set of the generated fuzzy rule is replaced by the *don't care* condition using a pre-specified probability $P_{don't\ care}$:

$$P(B_k) = \frac{\mu_{B_k}(x_{pi})}{\sum_{j=1}^{14} \mu_{B_j}(x_{pi})}. \tag{7}$$

- Fitness computation: The fitness value of each rule set S_i in the current population is calculated as the number of correctly classified training patterns by S_i. For the Michigan approach the computation follows the same scheme.
- Selection: It is based on binary tournament.
- Crossover: The substring-wise and bit-wise uniform crossover is applied in the Pittsburgh part. In the case of the Michigan part only the bit-wise uniform crossover is considered.
- Mutation: Each fuzzy partition of the individuals is randomly replaced by a different fuzzy partition using a pre-specified mutation probability for both approaches.

In our study, we have used the following parameters' values for the Ishibuchi and Yamamoto's FH-GBML method:

- Number of fuzzy rules: $5 \times p$ rules (where p is the number of examples in the data set).
- Number of rule sets (N_{pop}): 200 rule sets.
- Crossover probability: 0.9.
- Mutation probability: $1/p$ (where p is the number of examples in the data set).
- Number of replaced rules: All rules except the best-one (Pittsburgh-part, elitist approach), number of rules/5 (Michigan-part).
- Total number of generations: 1000 generations.
- Don't care probability: 0.5.
- Probability of the application of the Michigan iteration: 0.5.

For more details about this proposal, please refer to [18].

2.3. Genetic fuzzy systems

A GFS is basically a fuzzy system augmented by a learning process based on evolutionary computation, which includes genetic algorithms, genetic programming, and evolutionary strategies, among other evolutionary algorithms (EAs) [11].

The automatic definition of a fuzzy rule based system (FRBS) can be seen as an optimization or search problem. EAs are a well known and widely used global search technique with the ability to explore a large search space for suitable solutions only requiring a performance measure. In addition to their ability to find near optimal solutions in complex search spaces, the generic code structure and independent performance features of EAs make them suitable candidates to incorporate *a priori* knowledge. In the case of FRBSs, this *a priori* knowledge may be in the form of linguistic variables, fuzzy membership function parameters, fuzzy rules, number of rules, etc. These capabilities extended the use of EAs in the development of a wide range of approaches for designing FRBSs over the last few years, as has been pointed out in the last international journal special issues on GFSs [4,5,9,6].

Finally, an extensive review of the most recent developments of GFS and FRBS can be found in [12]. The web site http://sci2s.ugr.es/gfs/ provides complete information and material on the topic.

3. Data complexity measures

In the following subsections, we first present a short review on recent studies on data complexity metrics (Section 3.1), and then we describe the measures of overlapping (Section 3.2), measures of separability of classes (Section 3.3) and measures of geometry (Section 3.4) used in our study.

3.1. Recent studies on data complexity

As we have mentioned, data complexity measures are a series of metrics that quantify data set characteristics which imply some difficulty to the classification task. In the following we gather several recent publications related to these complexity measures and their applications. They can show a picture of the most recent developments in the topic:

- In [14], Ho and Basu propose some complexity measures for binary classification problems, gathering metrics of three types: overlaps in feature values from different classes; separability of classes; and measures of geometry, topology, and density of manifolds.
- In [23], Singh offers a review of data complexity measures and proposes two new ones.

Table 1
Complexity metrics used in this study.

Measure	Description
F2	Volume of overlap region
F3	Maximum (individual) feature efficiency
L1	Minimized sum of error distance by linear programming
L2	Error rate of linear classifier by linear programming
N2	Ratio of average intra/inter class NN distance
N3	Error rate of 1-NN classifier
N4	Nonlinearity of 1-NN classifier
T2	Average number of points per dimension

- In [3], Bernadó-Mansilla and Ho investigate the domain of competence of XCS by means of a methodology that characterizes the complexity of a classification problem by a set of geometrical descriptors.
- In [20], Li et al. analyse some omnivariate decision trees using the measure of complexity based in data density proposed by Ho and Basu.
- Baumgartner and Somorjai define specific measures for regularized linear classifiers in [2], using Ho and Basu's measures as reference.
- Sánchez et al. analyse the effect of the data complexity in the nearest neighbours (NNs) classifier in [22].
- Dong and Kothari propose in [10] a feature selection algorithm based on a complexity measure defined by Ho and Basu.
- Mollineda et al. in [21] extend some of Ho and Basu's measure definitions for problems with more than two classes. They analyse these generalized measures in two classic prototype selection algorithms and remark that Fisher's discriminant ratio is the most effective for prototype selection.

In our study we will study eight of the measures proposed in [14] which offer information for the FH-GBML method. They are summarized in Table 1.

In the following subsections we describe the measures we have used, classified by their family.

3.2. Measures of overlaps in feature values from different classes

These measures focus on the effectiveness of a single feature dimension in separating the classes, or the composite effects of a number of dimensions. They examine the range and spread of values in the data set within each class, and check for overlaps among different classes.

F2: Volume of overlap region. Let the maximum and minimum values of each feature f_i in class C_j be $\max(f_i, C_j)$ and $\min(f_i, C_j)$, then the overlap measure F2 is defined as

$$F2 = \prod_i \frac{MINMAX_i - MAXMIN_i}{MAXMAX_i - MINMIN_i},$$

where $i = 1, \ldots, d$ for a d-dimensional problem, and

$$MINMAX_i = MIN(\max(f_i, C_1), \max(f_i, C_2)),$$
$$MAXMIN_i = MAX(\min(f_i, C_1), \min(f_i, C_2)),$$
$$MAXMAX_i = MAX(\max(f_i, C_1), \max(f_i, C_2)),$$
$$MINMIN_i = MIN(\min(f_i, C_1), \min(f_i, C_2)).$$

F2 measures the amount of overlap of the bounding boxes of two classes. It is the product of per-feature overlap ratios, each of which is the width of the overlap interval normalized by the width of the entire interval encompassing the two classes. The volume is zero as long as there is at least one dimension in which the value ranges of the two classes are disjoint.

F3: Maximum (individual) feature efficiency. In a procedure that progressively removes unambiguous points falling outside the overlapping region in each chosen dimension [13], the efficiency of each feature is defined as *the fraction of all remaining points separable by that feature*. To represent the contribution of the most useful feature in this sense, we use the maximum feature efficiency (largest fraction of points distinguishable with only one feature) as a measure (F3). This measure considers only separating hyperplanes perpendicular to the feature axes. Therefore, even for a linearly separable problem, F3 may be less than 1 if the optimal separating hyperplane is oblique.

3.3. Measures of separability of classes

These measures give indirect characterizations of class separability. They assume that a class is made up of a single or multiple manifolds that form the support of the probability distribution of the given class. The shape, position and interconnectedness of these manifolds give hints on how well two classes are separated, but they do not describe separability by design. Some examples are shown as follows:

L1: Minimized sum of error distance by linear programming (LP). Linear classifiers can be obtained by a linear programming formulation proposed by Smith [24]. The method minimizes the sum of distances of error points to the separating hyperplane (subtracting a constant margin):

$$\begin{aligned} &\text{minimize} && \mathbf{a}^{\mathbf{t}}\mathbf{t} \\ &\text{subject to} && \mathbf{Z}^{\mathbf{t}}\mathbf{w} + \mathbf{t} \geqslant \mathbf{b}, \\ & && \mathbf{t} \geqslant \mathbf{0}, \end{aligned}$$

where $\mathbf{a}$, $\mathbf{b}$ are arbitrary constant vectors (both chosen to be 1), $\mathbf{w}$ is the weight vector to be determined, $\mathbf{t}$ is an error vector, and $\mathbf{Z}$ is a matrix where each column $\mathbf{z}$ is defined on an input vector $\mathbf{x}$ (augmented by adding one dimension with a constant value 1) and its class C (with value C_1 or C_2) as follows:

$$\begin{cases} \mathbf{z} = +\mathbf{x} & \text{if } C = C_1, \\ \mathbf{z} = -\mathbf{x} & \text{if } C = C_2. \end{cases}$$

The value of the objective function in this formulation is used as a measure (L1). The measure has a zero value for a linearly separable problem. Its value can be heavily affected by outliers located in the wrong side of the optimal hyperplane. The measure is normalized by the number of points in the problem and also by the length of the diagonal of the hyperrectangular region enclosing all training points in the feature space. It is zero for a linearly separable problem. We should notice that this measure can be heavily affected by the presence of outliers in the data set.

L2: Error rate of linear classifier by linear programming. This measure is the error rate of the linear classifier defined for L1, measured with the training set. With a small training set this can be a severe underestimate of the true error rate.

N2: Ratio of average intra/inter class nearest neighbour distance. For each input instance x_p, we calculate the distance to its nearest neighbour within the class ($intraDist(x_p)$) and the distance to nearest neighbour of any other class ($interDist(x_p)$). Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example, i.e.,

$$N2 = \frac{\sum_{i=0}^{m} intraDist(x_i)}{\sum_{i=0}^{m} interDist(x_i)},$$

where m is the number of examples in the data set. This metric compares the within-class spread with the distances to the nearest neighbours of other classes. Low values of this metric suggest that the examples of the same class lay closely in the feature space. Large values indicate that the examples of the same class are disperse. It is sensitive to the classes of the closest neighbours to a point, and also to the difference in magnitude of the between-class distances and that of the within-class distances.

N3: Error rate of 1-NN classifier. This is simply the error rate of a nearest neighbour classifier measured with the training set. The error rate is estimated by the leave-one-out method. The measure denotes how close the examples of different classes are. Low values of this metric indicate that there is a large gap in the class boundary.

3.4. Measures of geometry, topology, and density of manifolds

These measures evaluate to what extent two classes are separable by examining the existence and shape of the class boundary. The contributions of individual feature dimensions are combined and summarized in a single score, usually a distance metric, rather than evaluated separately. Two measures from this family are described as follows:

N4: Nonlinearity of 1-NN classifier. This is the nonlinearity measure, as defined by linear programming. Hoekstra and Duin [16] proposed a measure for the nonlinearity of a classifier with respect to a given data set. Given a training set, the method first creates a test set by linear interpolation (with random coefficients) between randomly drawn pairs of points from the same class. Then the error rate of the classifier (trained by the given training set) on this test set is measured. Here we use such a nonlinearity measure for the linear classifier defined for L1. In the case of N4, error is calculated for a nearest neighbour classifier. This measure is for the alignment of the nearest neighbour boundary with the shape of the gap or overlap between the convex hulls of the classes.

T2: Average number of points per dimension. This is a simple ratio of the number of points in the data set over the number of feature dimensions, i.e.,

$$T2 = \frac{m}{n},$$

where m is the number of examples in the data set and n is the number of attributes of the data set. This measure is included mostly for connection with prior studies on sample sizes. Because the volume of a region scales exponentially with the number of dimensions, a linear ratio between the two is not a good measure of sampling density.

4. Data sets choice for the experimental study

We evaluate FH-GBML on a set of 438 binary classification problems. These problems are generated from pairwise combinations of the classes of 21 problems from the University of California, Irvine (UCI) repository [1]. The selected ones are *iris*, *wine*, *new-thyroid*, *solar-flare*, *led7digit*, *zoo*, *yeast*, *tae*, *balanced*, *car*, *contraceptive*, *ecoli*, *hayes-roth*, *shuttle*, *australian*, *pima*, *monks*, *bupa*, *glass*, *haberman*, and *vehicle*.

In order to do that, first we take each data set and extract the examples belonging to each class. Then we construct a new data set with the combination of the examples from two different classes. This will result in a new data set with only two classes and the examples which have two such classes as output. For example, one data set obtained from *iris* with this procedure could contain only the examples of *Iris-setosa* and *Iris-virginica* and not those from *Iris-versicolor*.

We perform this process for every possible pairwise combination of classes. However, if a data set obtained with this procedure proves to be linearly separable, we discard it. If the data set proves to be linearly separable, then we could classify it with a linear classifier with no error, so such a data set would not be a representative problem. The complexity measure L1 indicates if a problem is linearly separable when its value is zero, so every data set with a L1 value of zero will be discarded.

This method for generating binary data sets is limited by the combinatorics itself, and we can only obtain over 200 new data sets with the original 20 data sets with this first approach. In order to obtain additional data sets, we take the next step to combine the classes from a data set: we group the classes two by two, that is, we create a new binary data set, and each of its two classes are the combination of two original classes each. For this second approach we have used *ecoli*, *glass*, and *flare* data sets, since they have a high number of class labels. For example, using *ecoli* we can create a new binary data set combining *cp* and *im* classes into a new one (say class "A"), and *pp* and *imU* (say new class "B"). The new data set would contain only the examples of classes *cp, im, pp*, and *imU*, but their class label now would be "A" if their original class was *cp* or *im*, and "B" if it was *pp* or *imU*. Again, those data sets with an L1 value of 0 are discarded.

Finally, these pairwise combinations resulted in 438 binary classification problems which are used as our test-bed. Although simple, these two methods for creating binary data sets produce a wide range of values for the complexity measures, even from the same original data set.

To estimate the classifiers' error, we use a 10-fold cross validation test once all the measures are computed. We take the mean accuracy of training and test of the 10 partitions as a representative measure of the method's performance.

Fig. 2 contains the results of FH-GBML showing the training and test accuracy over all the 438 data sets, plotted in ascending training accuracy value. We would like to point out how over-fitting is continuously present in Fig. 2.

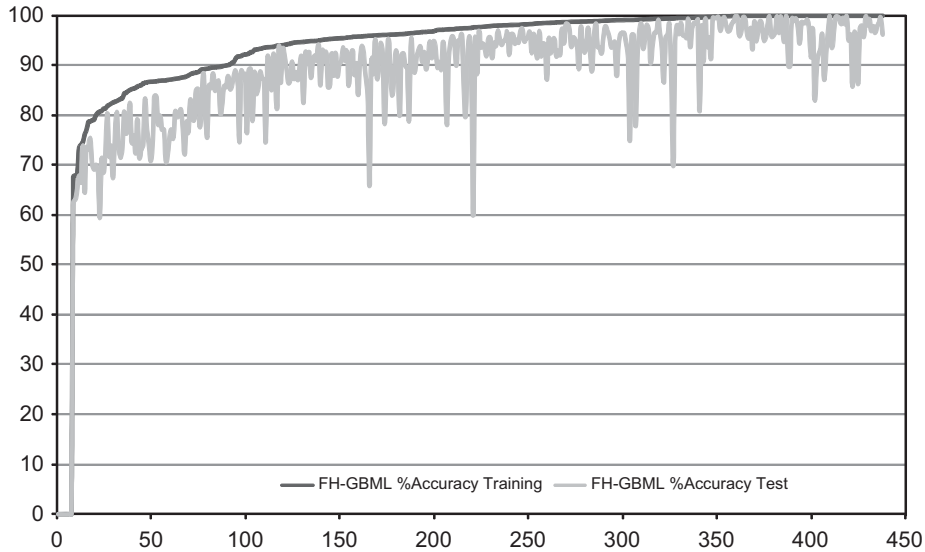

Fig. 2. Accuracy in training/test for FH-GBML sorted by training accuracy.

5. Experimental study: analysis of the FH-GBML method with data complexity measures

This study begins with the obtained results of the FH-GBML for the data sets considered. For each complexity measure, the data sets are sorted by its value, and put altogether in a figure. From these figures we obtain useful intervals which represent a good or bad behaviour of the classification method for the mentioned eight complexity measures. From these intervals we construct several rules that model the performance of the used FRBCS.

In order to do this analysis, we divide this section into the following two studies:

1. Determination of rules based on FH-GBML method's behaviour in Section 5.1.
2. Analysis of the collective evaluation of the set of rules in Section 5.2.

5.1. Determination of rules based on FH-GBML method's behaviour

First, we must point out what we understand for *good and bad behaviour* of FH-GBML:

- We understand for *good behaviour* an average high test accuracy in the interval as well as the absence of over-fitting.
- By *bad behaviour* we refer to the presence of over-fitting and/or average low test accuracy in the interval.

In the following we present the results of the execution over the 438 data sets summarized in Figs. 3–10. In each figure the results obtained by the FH-GBML method are sorted by the ascending value of the corresponding complexity measure. In the X axis we represent the data sets, not the complexity measure value, and the Y axis depicts the accuracy obtained both in training and test. The reason to do so is to give each data set the same space in the graphic representation. For those measures where we can find different *ad hoc* intervals which present good or bad behaviour of the FH-GBML, we use a vertical line to delimit the interval of the region of interest.

In Table 2 we have summarized the intervals found *ad hoc* from Figs. 3–10.

Once we have defined the *ad hoc* intervals, in Table 3 we have summarized the rules derived from them. Given a particular data set X, we get the complexity measure (CM) of X with the notation $CM[X]$. Table 3 is organized with the following columns:

- The first column corresponds to the identifier of the rule for further references.
- The "Rule" column presents the rule itself.
- The third column "Support" presents the percentage of data sets which verify the antecedent part of the rule.

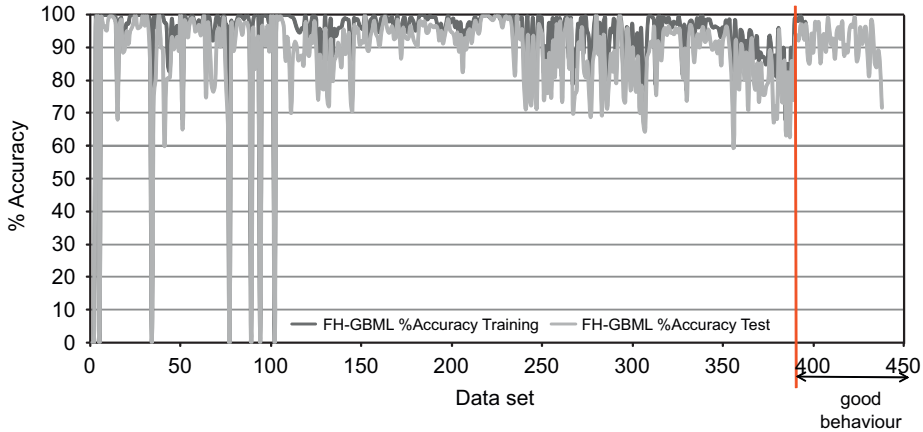

Fig. 3. Accuracy in training/test sorted by F2.

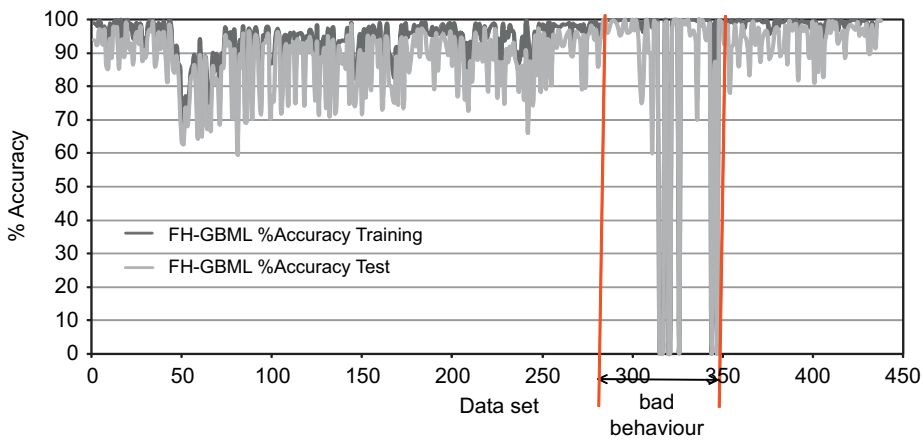

Fig. 4. Accuracy in training/test sorted by F3.

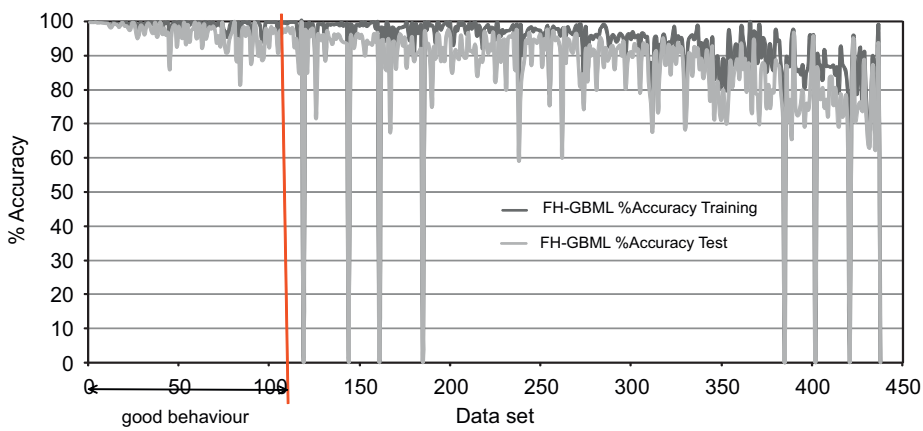

Fig. 5. Accuracy in training/test sorted by N2.

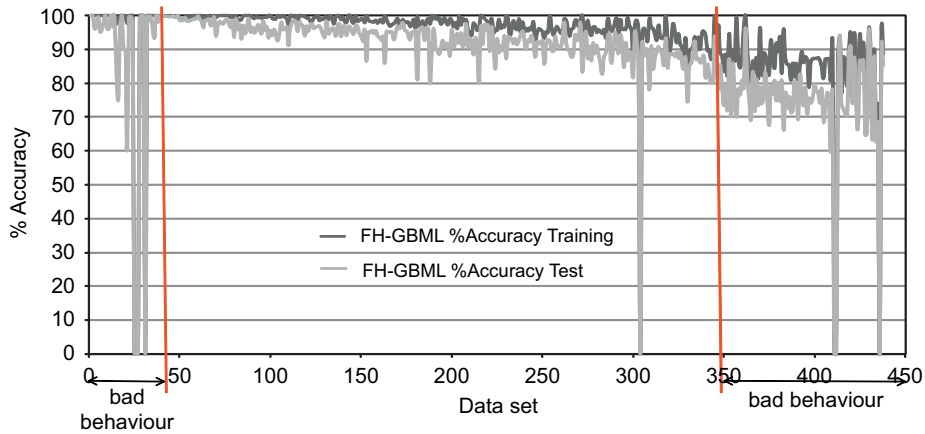

Fig. 6. Accuracy in training/test sorted by N3.

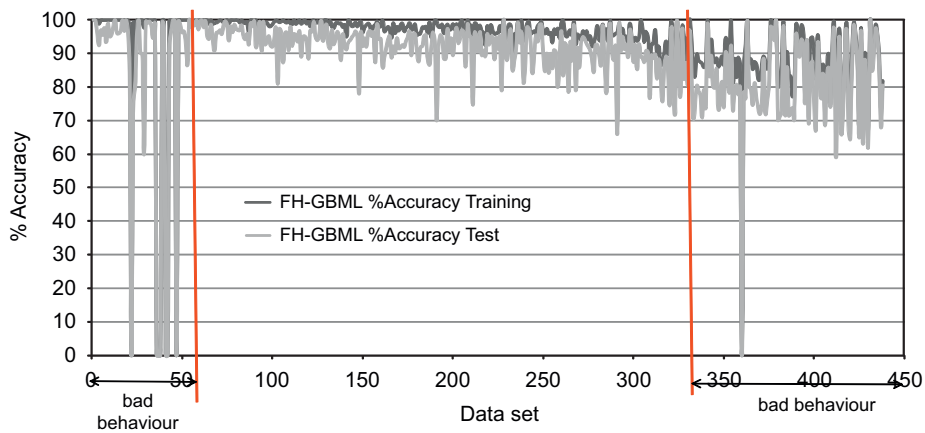

Fig. 7. Accuracy in training/test sorted by N4.

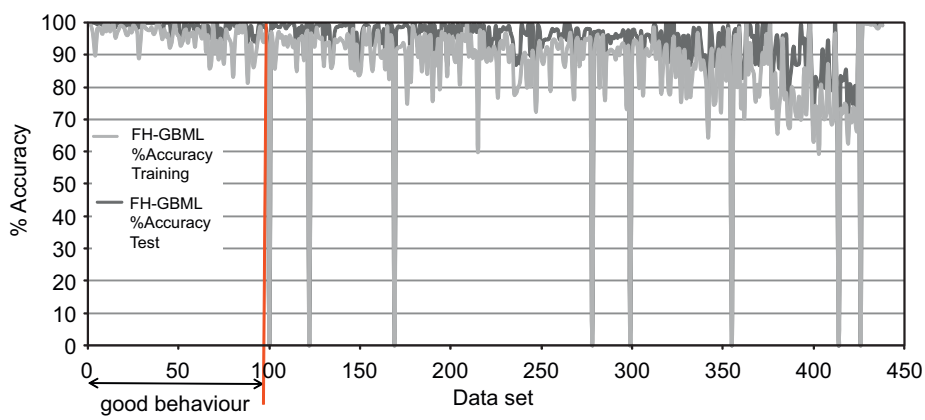

Fig. 8. Accuracy in training/test sorted by L1.

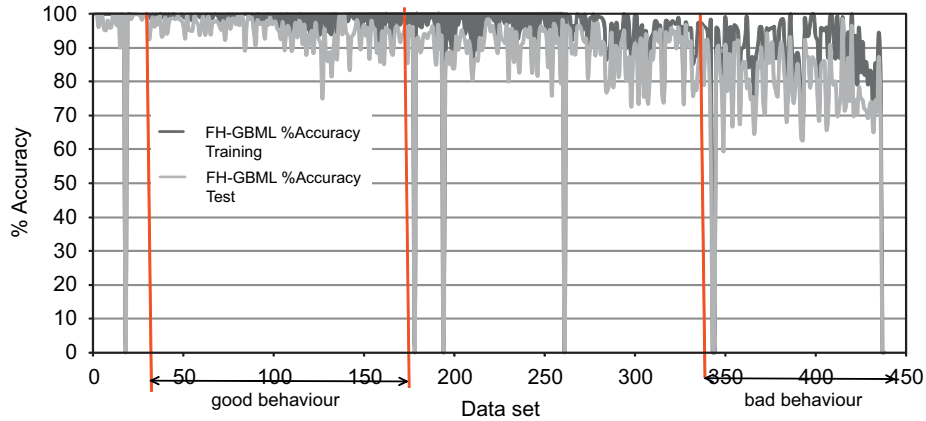

Fig. 9. Accuracy in training/test sorted by L2.

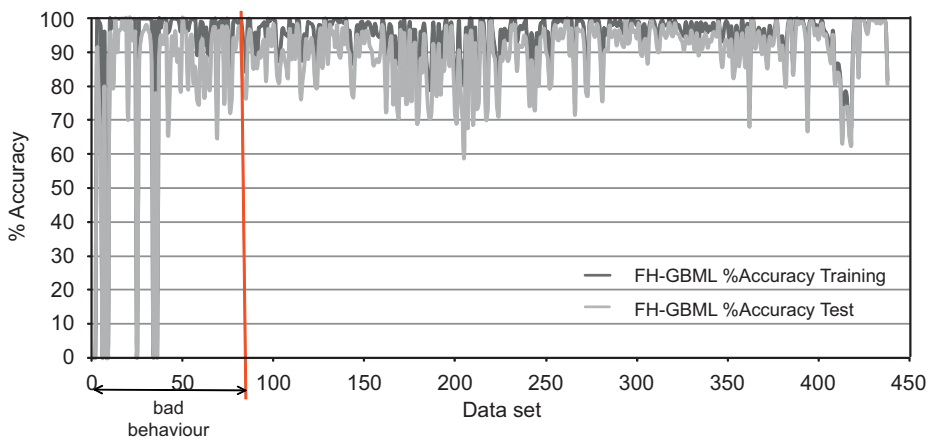

Fig. 10. Accuracy in training/test sorted by T2.

Table 2
Significant intervals.

Interval	FH-GBML behaviour
$N2 < 0.23$	*good behaviour*
$L1 < 0.2231$	*good behaviour*
$F2 = 1$	*good behaviour*
$L2 < 0.125$	*good behaviour*
$N3 = 0$	*bad behaviour*
$N3 > 0.1631$	*bad behaviour*
$N4 = 0$	*bad behaviour*
$N4 > 0.1743$	*bad behaviour*
$1 < F3 < 1.83$	*bad behaviour*
$L2 > 0.2834$	*bad behaviour*
$T2 < 12.29$	*bad behaviour*

- The column "% training, Std. Dev." shows the average accuracy in training of all the examples which are covered by the rule. The standard deviation of the average training accuracy is computed as well.
- The column "Training diff." contains the difference between the training accuracy of the rule and the training accuracy across all 438 data sets.

Table 3
One metric rules with obtained from the intervals.

Id.	Rule	Support (%)	% training, Std. Dev.	Training diff. (%)	% test, Std. Dev.	Test diff. (%)
R1+	If $N2[X] < 0.23$ then *good behaviour*	25.342	99.283, 1.340	5.713	96.854, 3.371	8.612
R2+	If $L1[X] < 0.2231$ then *good behaviour*	22.603	98.764, 1.868	5.195	95.754, 3.868	7.512
R3+	If $F2[X] = 1$ then *good behaviour*	11.187	96.060, 4.050	2.490	91.829, 5.475	3.588
R4+	If $L2[X] < 0.125$ then *good behaviour*	35.616	98.388, 2.271	4.818	95.094, 4.176	6.852
R1−	If $1 <$ F3[X] < 1.83 then *bad behaviour*	16.210	88.480, 31.537	−5.090	84.305, 30.937	−3.937
R2−	If $N3[X] = 0$ then *bad behaviour*	8.676	89.325, 30.643	−4.244	85.460, 30.272	−2.782
R3−	If $N3[X] > 0.1631$ then *bad behaviour*	21.005	83.531, 16.633	−10.038	74.521, 15.463	−13.721
R4−	If $N4[X] = 0$ then *bad behaviour*	12.557	87.083, 33.262	−6.487	82.941, 32.390	−5.301
R5−	If $N4[X] > 0.1743$ then *bad behaviour*	24.201	87.250, 11.239	−6.319	80.741, 12.707	−7.501
R6−	If $L2[X] > 0.2834$ then *bad behaviour*	22.146	85.917, 19.422	−7.652	76.114, 18.050	−12.128
R7−	If $T2[X] < 12.29$ then *bad behaviour*	17.580	87.7356, 30.143	−5.834	79.431, 28.609	−8.811

Table 4
Average FH-GBML training and test accuracy.

FH-GBML global % accuracy training	93.56955
FH-GBML global % accuracy test	88.24174

- The column "% test, Std. Dev." shows the average accuracy in test of all the examples which are covered by the rule. The standard deviation of the average test accuracy is computed as well.
- The column "Test diff." contains the difference between the test accuracy of the rule and the test accuracy across all 438 data sets.

In Table 4 we show the global training and test accuracy obtained by the FH-GBML method across all 438 data sets.

As we can see in Table 3, the positive rules (denoted with a "+" symbol in their identifier) always show a positive difference with the global average, both in training and test accuracy. The negative ones (with a "−" symbol in their identifier) verify the opposite case. The support of the rules shows us that we can characterize a wide range of data sets and obtain significant differences in accuracy, as we can see from rules R1+, R2+, R4+, R3− and R5−. Notice also that the standard deviations for the positive rules are always lower than those for the negative ones. The robustness of the method is higher in those data sets which have been covered by a positive rule. FH-GBML performs worse in the data sets under a negative rule, due usually to method's over-fitting.

From this set of rules we can state that:

- FH-GBML performs well in those data sets in which the examples of the same class lay closely in the feature space, since a low N2 offers *good* results.
- When the problem is linear or almost linearly separable, we obtain *good* results, as can be drawn from low values of L1.

Table 5
Disjunction and intersection rules from all simple rules.

Id.	Rule	Support (%)	% training, Std. Dev.	Training diff. (%)	% test, Std. Dev.	Test diff.
PRD	If R1+ or R2+ or R3+ or R4+ *good behaviour*	42.694	98.360, 3.241	4.791	95.170, 6.213	6.929
NRD	If R1− or R2− or R3− or R4− or R5− or R6− or R7− *bad behaviour*	55.479	90.325, 18.168	−3.245	83.864, 18.46784	−4.378
PRD∧ NRD	If PRD and NRD then *good behaviour*	22.602	98.319, 3.316	4.750	95.048, 5.424	6.806
PRD∧¬NRD	If PRD and not NRD then *good behaviour*	20.091	98.406, 1.837	4.837	95.308, 3.455	7.067
NRD∧¬PRD	If NRD and not PRD then *bad behaviour*	32.877	84.829, 21.801	−8.741	76.175, 20.254	−12.066
Not characterized	If not PRD and not (NRD and not PRD) then *good behaviour*	24.429	96.960, 2.097	3.391	92.372, 3.036	4.130

- A low error rate of the LP classifier in the classification, defined by L1 and measured by L2, results in an *acceptable good* behaviour of FH-GBML. On the other hand, when the error rate of the LP classifier grows sufficiently, the behaviour of FH-GBML becomes *bad* with a difference of −12.13%.
- When the volume of overlap (F2) is 1, the FH-GBML obtains *acceptable good* results with an improvement of 6.85% with respect to the global average.
- When the error obtained by the 1-NN method (N3) is 0, the FH-GBML method usually does not perform well as well as when the nonlinearity of the 1-NN method (N4) is 0. With these conditions, FH-GBML obtains a *relative bad* behaviour with differences of −2.78% and −5.30%, respectively.
- The parallel behaviour of N3 and N4 measures is present when they have high values as well. In particular when the error obtained by the 1-NN method (N3) is above 0.1631 and when the non-linearity of the 1-NN method (N4) is above 0.1743. With these conditions, FH-GBML obtains a *bad* behaviour with differences of −13.72% and −7.50%, respectively.
- An interesting fact is that a ratio between the number of examples and the number of attributes (T2) lower than 12.29 generally results in *bad* performance.

Although we have obtained some interesting rules, we can extend our study by considering the combination of these complexity metrics in order to obtain more precise and descriptive rules.

5.2. Collective evaluation of the set of rules

The objective of this section is to jointly analyse the good rules, as well as the bad rules. Thus we can arrive at a more general description, with a wider support, of the behaviour of the FH-GBML method with these joint rules. We perform the disjunctive combination of all the positive rules to obtain a single rule. The same is done with all the negative ones, so we obtain another rule. The new disjunctive rule will be activated if any of the component rules' antecedents are verified.

Since the support of the joint rules will be high, we also compute the intersection of these disjunctive rules (the data sets which activate both disjunctive rules). With the intersection of the disjunction, we try to see the global relations and competence between positive and negative intervals.

Thus we obtain three different kinds of intersections:

- Intersection of positive disjunction and *not* the negative disjunction.
- Intersection of positive disjunction and the negative disjunction.
- Intersection of negative disjunction and *not* the positive disjunction.

In Table 5 we summarize both disjunctions and the three intersections.

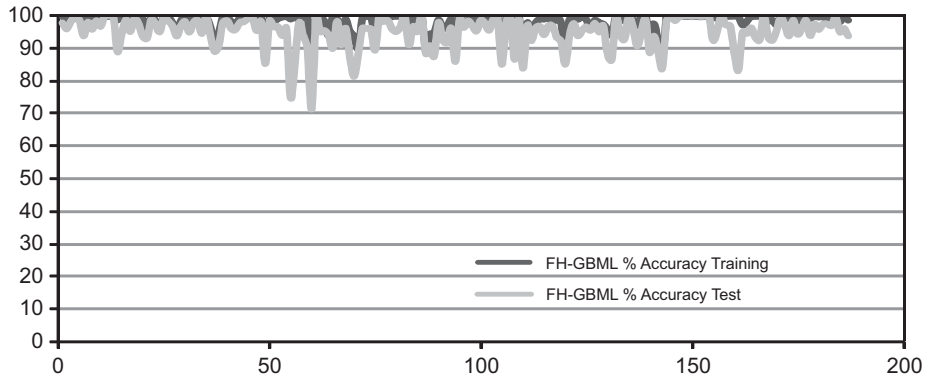

Fig. 11. Accuracy results for data sets covered by PRD.

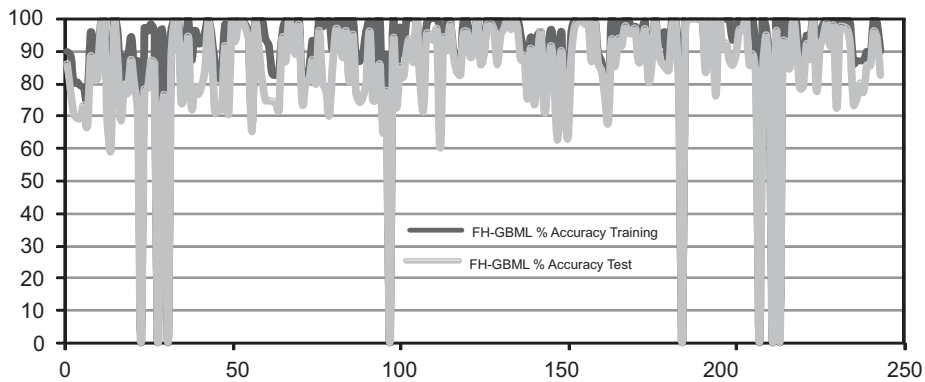

Fig. 12. Accuracy results for data sets covered by NRD.

From the new obtained rules, we can point out that:

- The positive rule disjunction (PRD) offers a high support (almost the half of the considered data sets), and it gives a good test accuracy (over the 95%). In spite of its wide support, it presents a low standard deviation of 6.21. Fig. 11 shows FH-GBML's results for data sets covered by this rule.
- The negative rule disjunction (NRD) obtains a wide support as well (over the 50%). However, it is not very accurate in indicating the data sets with low FH-GBML methods's performance as we can see from its high standard deviation and low difference. Fig. 12 shows FH-GBML's results for the data sets included in the support of this rule.
- The positive and negative rule disjunction (PRD$\wedge$NRD) is more specific than PRD in isolation. However, it presents a similar standard deviation. It is also similar to PRD in the training and test accuracy difference. Fig. 13 shows the FH-GBML accuracy results of the data sets covered by this latter rule. The Positive and Not Negative Rule Disjunction (PRD$\wedge\neg$NRD) has a lower support than PRD$\wedge$NRD and a lower standard deviation, but its difference is somewhat higher, since the data sets with low accuracy for FH-GBML have been removed by PRD$\wedge$NRD. Fig. 14 shows the accuracy results of FH-GBML for the data sets covered by this rule.
- The negative and not positive rule disjunction (NRD$\wedge\neg$PRD) is a good rule to describe the bad behaviour of FH-GBML. It has a decent support and both a high difference in training and test sets. Fig. 15 shows the FH-GBML accuracy results for the data sets in the support of this rule.

From all these new rules, we can present PRD as a representative description of good data sets, and NRD$\wedge\neg$PRD as a representative description for bad data sets, when using FH-GBML. We can consider three blocks of data sets with

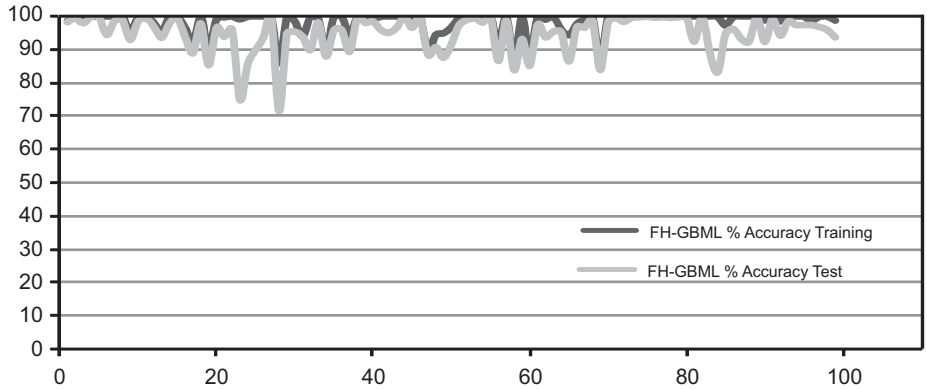

Fig. 13. Accuracy results for data sets covered by PRD∧NRD.

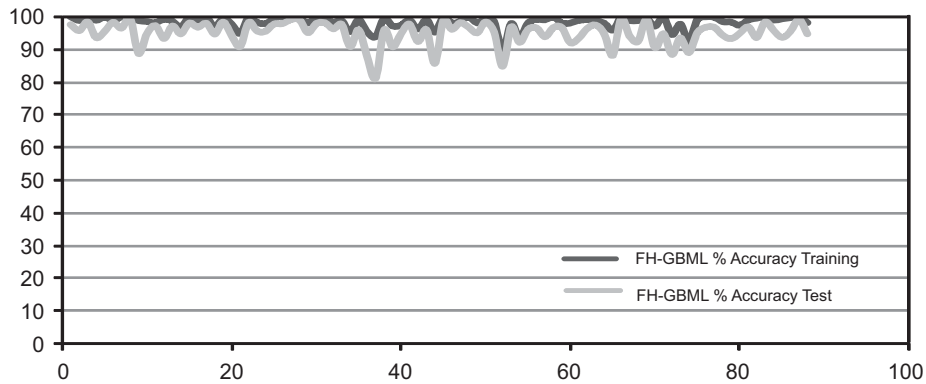

Fig. 14. Accuracy results for data sets covered by PRD∧¬NRD.

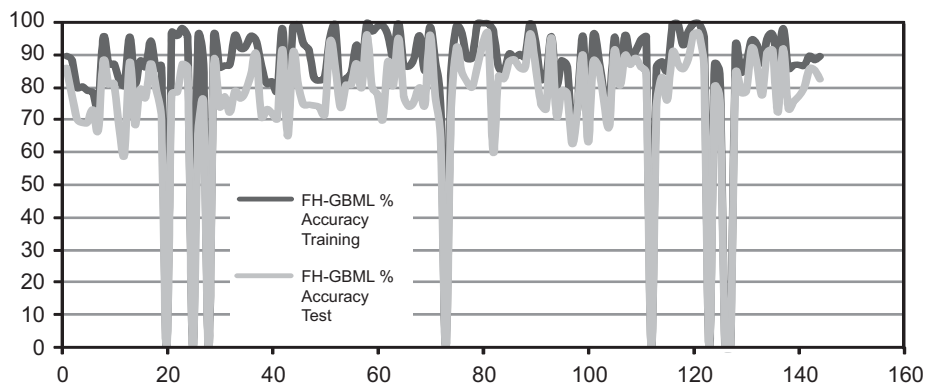

Fig. 15. Accuracy results for data sets covered by NRD∧¬PRD.

their respective support, as depicted in Fig. 17 (with no particular data set order within each block):

- The first block (the left-side one) represents those data sets covered by the PRD rule. They are the data sets recognized as being those in which FH-GBML has good accuracy.
- The second block (the middle one) plots the data sets for the rule NRD∧¬PRD, which are bad data sets for FH-GBML.

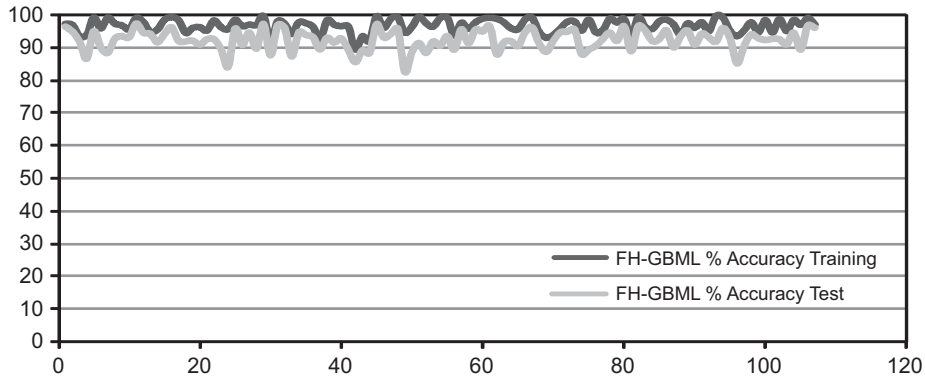

Fig. 16. Accuracy results for data sets not covered either by PRD and NRD∧¬PRD.

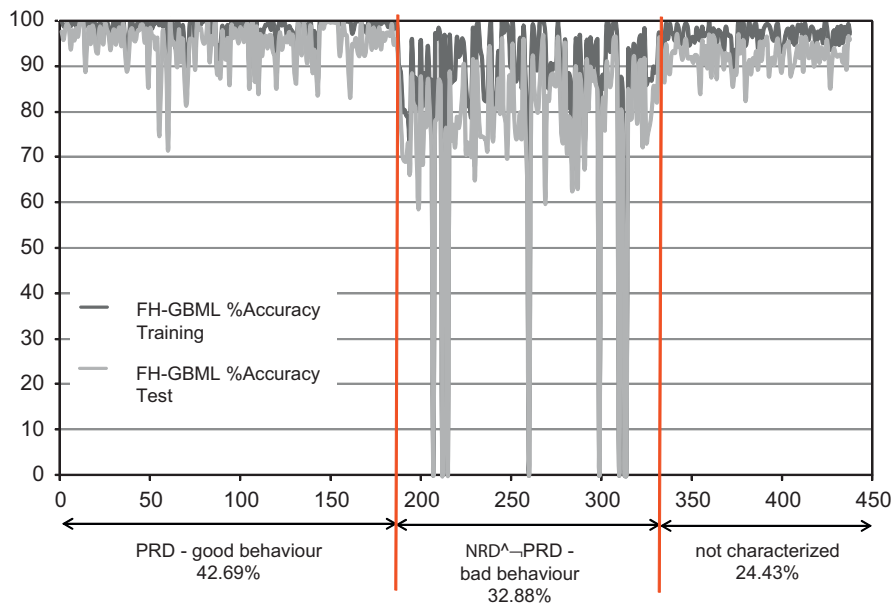

Fig. 17. Three blocks representation for PRD, NRD∧¬PRD and not covered data sets.

- The third and last block (the right-side one) contains the unclassified data sets by the previous two rules. This uncovered bunch of data sets is represented in the last row of Table 5. Fig. 16 also depicts these data sets.

We can see that the 75% of the analysed data sets are covered by these two rules, and hence the *good behaviour* and *bad behaviour* consequents properly represent the accuracy obtained by the FH-GBML method.

6. Conclusions

We have performed a study over a set of binary data sets with the FH-GBML method. We have computed some data complexity measures for the data sets in order to obtain intervals of such metrics in which the method's performance is significantly good or bad. We have constructed descriptive rules as well as studied the interaction between the intervals and the proper rules.

We have obtained two rules which are simple, interpretable and precise to describe both good and bad performance of the FH-GBML method. Furthermore, we present the possibility of determining which data sets would be prove to FH-GBML to perform well or badly prior to their execution, using the Data Complexity measures.

We must point out that this is a particular study for one specific method, the FH-GBML. On the other hand, this work presents a new challenge that could be extended to other FRBCSs, to analyse their domains of competence, and to develop new measures which could provide us with more information on the behaviour of FRBCSs for pattern recognition.

References

[1] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2007 ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩.

[2] R. Baumgartner, R.L. Somorjai, Data complexity assessment in undersampled classification, Pattern Recognition Letters 27 (2006) 1383–1389.

[3] E. Bernadó-Mansilla, T.K. Ho, Domain of competence of XCS classifier system in complexity measurement space, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 82–104.

[4] B. Carse, A.G. Pipe, Introduction: genetic fuzzy systems, International Journal of Intelligent Systems 22 (9) (2007) 905–907.

[5] J. Casillas, F. Herrera, R. Pérez, M.J. del Jesus, P. Villar, Special issue on genetic fuzzy systems and the interpretability—accuracy trade-off—editorial, International Journal of Approximate Reasoning 44 (1) (2007) 1–3.

[6] J. Casillas, B. Carse, Preface: genetic fuzzy systems: recent developments and future directions, Special issue on genetic fuzzy systems: recent developments and future directions, Soft Computing 13 (2009) 417–418.

[7] O. Cordón, M.J. del Jesus, F. Herrera, A proposal on reasoning methods in fuzzy rule-based classification systems, International Journal of Approximate Reasoning 20 (1) (1999) 21–45.

[8] O. Cordón, F. Gomide, F. Herrera, F. Hoffmann, L. Magdalena, Ten years of genetic fuzzy systems: current framework and new trends, Fuzzy Sets and Systems 141 (1) (2004) 5–31.

[9] O. Cordón, R. Alcalá, J. Alcalá-Fdez, I. Rojas, Genetic fuzzy systems, Special issue on genetic fuzzy systems: What's next?—editorial, IEEE Transactions on Fuzzy Systems 15 (4) (2007) 533–535.

[10] M. Dong, R. Kothari, Feature subset selection using a new definition of classificability, Pattern Recognition Letters 24 (2003) 1215–1225.

[11] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computation, Springer, Berlin, 2003.

[12] F. Herrera, Genetic fuzzy systems: taxonomy, current research trends and prospects, Evolutionary Intelligence 1 (2008) 27–46.

[13] T.K. Ho, H.S. Baird, Pattern classification with compact distribution maps, Computer Vision and Image Understanding 70 (1) (1998) 101–110.

[14] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 289–300.

[15] M. Basu, T.K. Ho, Data Complexity in Pattern Recognition, Springer, Berlin, 2006.

[16] A. Hoekstra, R.P.W, Duin, On the nonlinearity of pattern classifiers, in: Proc. 13th Internat. Conf. on Pattern Recognition, Vienna, 1996, pp. 271–275.

[17] H. Ishibuchi, T. Nakashima, M. Nii, Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining, Springer, Berlin, 2004.

[18] H. Ishibuchi, T. Yamamoto, T. Nakashima, Hybridization of fuzzy GBML approaches for pattern classification problems, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 35 (2) (2005) 359–365.

[19] L. Kuncheva L, Fuzzy Classifier Design, Springer, Berlin, 2000.

[20] Y. Li, M. Dong, R. Kothari, Classificability-based omnivariate decision trees, IEEE Transactions on Neural Networks 16 (6) (2005) 1547–1560.

[21] R.A. Mollineda, J.S. Sánchez, J.M. Sotoca, Data characterization for effective prototype selection, in: First Edition of the Iberian Conf. on Pattern Recognition and Image Analysis (IbPRIA 2005), Lecture Notes in Computer Science, vol. 3523, Springer, Berlin, 2005, pp. 27–34.

[22] J.S. Sánchez, R.A. Mollineda, J.M. Sotoca, An analysis of how training data complexity affects the nearest neighbor classifiers, Pattern Analysis and Applications 10 (3) (2007) 189–201.

[23] S. Singh, Multiresolution estimates of classification complexity, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (12) (2003) 1534–1539.

[24] F.W. Smith, Pattern classifier design by linear programming, IEEE Transactions on Computers 17 (4) (1968) 367–372.

2.2. Shared Domains of Competence of Approximative Models using Measures of Separability of Classes

- J. Luengo, F. Herrera, Shared Domains of Competence of Approximative Models using Measures of Separability of Classes. **Submitted to Information Sciences**.

  - Status: **Submitted**.
  - Impact Factor (JCR 2009): 3.291.
  - Subject Category: Computer Science, Information Systems. Ranking 6 / 116.

Elsevier Editorial System(tm) for Information Sciences
Manuscript Draft

Manuscript Number:

Title: Shared Domains of Competence of Approximative Models using Measures of Separability of Classes

Article Type: Full length article

Keywords: Classification; Data Complexity; Domains of Competence; Multi
Layer Perceptron; Radial Basis Function Network; Support Vector Machine; Learning Vector Quantization

Corresponding Author: Mr. Julian Luengo, M.D.

Corresponding Author's Institution: University of Granada

First Author: Julian Luengo, M.D.

Order of Authors: Julian Luengo, M.D.; Francisco Herrera, Full professor

Suggested Reviewers: Ester Bernado
esterb@salle.url.edu

Yusuke Nojima
nojima@cs.osakafu-u.ac.jp

Luciano Sanchez
luciano@uniovi.es

Shared Domains of Competence of Approximative Models using Measures of Separability of Classes☆

Julián Luengo[a,*], Francisco Herrera[a]

[a]*Dept. of Computer Science and Artificial Intelligence, CITIC-UGR, 18071, Granada, Spain.*

Abstract

In this work we jointly analyze the behavior of three classic Artificial Neural Network models and one Support Vector Machine with respect to a series of data complexity measures known as measures of separability of classes. In particular, we consider a Radial Basis Function Network, a Multi-Layer Perceptron, a Learning Vector Quantization, while the Sequential Minimal Optimization method is used to model the Support Vector Machine.

We consider five measures of separability of classes over a wide range of data sets built from real data which have proved to be very informative when analyzing the performance of classifiers. We find that two of them allow us to extract common behavior patterns for the four learning methods due to their related nature. We obtain rules using these two metrics that describe both good or bad behaviors of the Artificial Neural Networks and the Support Vector Machine.

With the obtained rules, we characterize the behavior of the methods from the data set complexity metrics and therefore their common domains of competence are established. Using these domains of competence the shared good and bad capabilities of these four models can be used to know if the approximative models will perform well or poorly or if a more complex configuration of the model is needed for a given problem in advance.

Keywords: Classification, Data Complexity, Domains of Competence, Multi Layer Perceptron, Radial Basis Function Network, Support Vector Machine, Learning Vector Quantization

☆Supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01. J. Luengo holds a FPU scholarship from Spanish Ministry of Education and Science.

*Corresponding author

Email addresses: julianlm@decsai.ugr.es (Julián Luengo), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

The use of Artificial Neural Networks (ANNs) in several tasks and fields is very common nowadays. Due their excellent adjusting capabilities, they have been successfully applied in the Data Mining ambit among many others[29], becoming a referent. In particular, ANNs have been succeeded in the classification task, within the Data Mining field. More recently, Support Vector Machines (SVMs) have become very popular in the same task due their high precision capabilities and their fast learning rate[10]. They share some common properties with respect to the ANNs, and it is usual to find them related in the specialized literature.

The research on ANNs in classification is very active nowadays as there are many recent contributions in this topic. Improvements in the weight adjusting for MLP in classification is a recurrent topic[1], even in challenging scenarios like imbalanced data[11, 33]. RBFNs have been applied to incremental learning[28] and their adjustment is subject to recent enhancements[13]. Villmann et al.[35] exploit the synergy between fuzzy classification models and Learning Vector Quantization (LVQ) models. We can find from extensive comparisons of the LVQ models[16] to the study of the initial values in the MLP backpropagation scheme[38], including improvements on the RBFN learning methodology based on kernels[25]. Even the use of non-parametric statistics in the classification framework have been studied for MLPs and RBFNs models[20].

The use of SVMs is also very prolific in the same topic. Recent contributions include new developments in classification assuming fixed probability distributions of the data[36], the use of recursive SVMs to tackle the dimensionality problem[34], the study of formulations of the loss function in order to deal with imbalanced data sets directly[37] or simultaneously selects relevant features during classifier construction [22]. SVMs and ANNs are very related in their foundations and their integration has been already studied [9], and the analysis on the different properties they model and their advantages have been studied [14].

It is well-known that the prediction capabilities of ANNs and SMVs in classification are dependent on the problem's characteristics. Moreover determining the adequate topology or parameters of these models is usually determined by the intrinsic characteristic of the data set and it is difficult to estimate them a priori. An emergent field, that uses a set of data complexity measures applied to the problem to try to capture different aspects or sources of complexity which are considered complicated to the classification task[3]. Studies of data complexity metrics applied to particular classification's algorithms can be found[5, 4, 30, 21].

In this work we are interested in to determine the domains of competence of ANNs and SVMs using a particular set of metrics formerly known as *measures of separability of classes*[3]. We consider three classic models of ANNs, RBFN, MLP and LVQ; and the well-known Sequential Minimal Optimization (SMO) as a representative SVM model solving procedure. We want to describe and to analyze the shared strengths and weakness of these four models, by means

of establishing their domains of competence. The measures of separability of classes in the data can be used to characterize this shared behavior, as they have proven to be very informative describing the domains of competence of rule-based classification methods[5, 21].

In order to perform this analysis, we create several binary classification data sets from real world problems, 438 in total, and compute the value of 5 measures of separability of classes proposed by Ho and Basu[17]. We have analyzed the intervals of measure values related to the created data sets, in which all the four methods perform well or bad, and then formulated a rule for such intervals. The obtained rules describe the ranges where some information and conclusions about the performance of ANN methods can be stated. Therefore the domains of competence[21] of the learning method are defined, that is, the intervals of the measures where the learning method performs well or badly. The generalization abilities of these domains of competence are checked using a fresh bunch of 200 data sets.

The rest of this paper is organized as follows. In Section 2 we briefly describe the ANNs and the SVM models used. In Section 3 the considered complexity metrics are summarized as well as the state-of-the-art literature in the topic. In Section 4 we show the process used to build up the bunch of data sets used and the parameter adjusting process for the four models. Next, in Section 5 we introduce the motivation for the use of the measures of separability of classes and the analysis related to the extraction of the learning method's domains of competence. Finally, in Section 6 some concluding remarks are pointed out.

2. Preliminaries: ANN and SVM details

In this section, we will briefly describe the most important details of the particular implementations of ANN and SVM models that we have used. In Subsection 2.1 the Multi-layer perceptron model used is presented. Next in Subsection 2.2 the details of the RBFN model are described. Subsection 2.3 depicts the details of the LVQ model used. Finally, in Subsection 2.4 the SVM used is described. All the models described in this section are implemented in the KEEL software package[1]. Please refer to the associated references in order to obtain further details.

2.1. Multi-Layer Perceptron

A Multi-Layer Perceptron[23] is a classical model which consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in a layer has directed connections to the neurons of the subsequent layer. The units of these networks apply a sigmoid function as an activation function. Each connection of the units are related to a weight. Therefore we

[1] `http://keel.es`

can define a weight vector as vector in the real euclidean space $\mathbb{R}^N$ where N is the number of weights and biases in the network.

$$\overrightarrow{w} = (\ldots, x_{ij}^l, w_{i+1j}^l, \ldots) \tag{1}$$

where w_{ij}^l is the weight from unit number i in layer number l to unit number j in layer number $l+1$.

The neural network has attached a global error function $E(\overrightarrow{w})$ depending on all weights and biases, for example the standard least square function, and its gradient descendent $E'(\overrightarrow{w})$.

There are many algorithms for feedforward neural networks, like the standard back propagation algorithm. Most of the optimization methods used to minimize the error function are a local iterative process in which an approximation to the function in a neighborhood of the current point in the space is minimized, given by a first or second order Taylor expansion of the function. First a initial weight vector $\overrightarrow{w}_1$ is chosen. Then a search direction $\tilde{p}_k$ and a step size α_k are determined in step k so that $E(\overrightarrow{w}_k + \alpha_k \tilde{p}_k) < E(\overrightarrow{w}_k)$. If the final condition is not fulfilled then the weight vector is updated $\overrightarrow{w}_{k+1} = \overrightarrow{w}_k + \alpha_k \tilde{p}_k$ and a new step $k+1$ begins.

However, more recent and scalable approaches can be used and with better convergence. The scaled conjugate gradient (SCG) avoids the line search per learning iteration. In the case of the back propagation algorithm, the seach direction $\tilde{p}_k$ is set to the negative gradient $-E'(\overrightarrow{w})$ and the step size α_k to a constant ε, considering the linear approximation $E(\overrightarrow{w}+\overrightarrow{y}) \approx E(\overrightarrow{w}) + E(\overrightarrow{w})^T y$. An attempt to improve the convergence rate for back propagation is the addition of the momentum term, but also increases the user-dependant parameters.

The SCG employs the above general optimization strategy but choose the search direction more carefully by using information from the second order approximation

$$E(\overrightarrow{w} + \overrightarrow{y}) \approx E(\overrightarrow{w}) + E(\overrightarrow{w})^T \overrightarrow{y} + \frac{1}{2} y^T E''(\overrightarrow{w}\,\overrightarrow{y}). \tag{2}$$

The step size $\tilde{s}_k = E''(\overrightarrow{w}_k)\tilde{p}_k$ is estimated as

$$\tilde{s}_k = E''(\overrightarrow{w}_k)\tilde{p}_k \approx \frac{E'(\overrightarrow{w}_k + \sigma_k \tilde{p}) - E'(\overrightarrow{w}_k)}{\sigma_k} + \lambda_k \tilde{p}_k.$$
$$0 < \sigma_k \ll 1. \tag{3}$$

which tends in the limit to the true value of $E''(\overrightarrow{w}_k)\tilde{p}_k$, and λ_k is adjusted in each iteration depending on the sign of $\delta_k = \tilde{p}_k^T \tilde{s}_k$. If $\lambda_k \leq 0$ then the Hessian matrix $E''(\overrightarrow{w})$ is not positive definite and λ_k is raised and $\tilde{s}_k$ estimated again. The λ_k value scales the step size en each iteration, giving the name to the algorithm.

2.2. Radial Basis Function Network

Radial Basis Function Networks[7, 8] are well suited for function approximation and pattern recognition due to their simple topological structure and

their ability to reveal how learning proceeds in an explicit manner. A RBF is a function which has been built into a distance criterion with respect to a centre. Different basis functions like thin-plate spline functions, multiquadratic functions, inverse multiquadratic functions and Gaussian functions have been proposed for the hidden-layer neurons, but normally the selected one is the Gaussian function. Compared with other types of Artificial Neural Networks, such as feed-forward networks, the RBFN requires less computation time for learning and also has a more compact topology. RBFs have been applied in the area of ANNs where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in multi-layer perceptrons. The original RBF method has been traditionally used for strict multivariate function interpolation [27] and for this fact, it requires as many RBF neurons as data points. Broomhead and Lowe removed this strict interpolation restriction[7] and provided a neural network architecture where the number of RBF neurons can be far less than the data points. A RBFN mainly consists of two layers, one hidden-layer and one output layer.

Each input example $x \in X$ is applied to all hidden neurons. The neuron i computes the function

$$h_i(x) = exp\left[\frac{(x - v_i)^2}{2\sigma_i^2}\right] \tag{4}$$

where v_i is the center of the neuron i, and h_i the output of such neuron. The RBFN has only one output (i.e. j has only one value), and it is defined by

$$z(x) = \frac{\sum_i^k h_i(x) w_i}{\sum_i^k h_i(x)} \tag{5}$$

The term w_i refers to the neuron weight, and k is the number of neurons in the hidden layer. In order to initialize the neurons in the network, we use a K-means clustering algorithm. The centre of the neuron is set equal to the centroid of the cluster, and its radius equal to the mean of the distance between the given center and the $N = 2$ nearest neurons:

$$\sigma_i = \sum_{j=1}^{N} \frac{d(v_i, v_j)}{N} \tag{6}$$

The network is initialized with a K-means clustering algorithm, setting the number of clusters equal to the number of neurons. The initial centroids are set to random chosen examples, which are all different. By successively iterations, the neurons are adjusted using the Euclidean distance till the centroids (i.e. the neurons) do not change.

Once the centers and radius of the neurons has been initialized, the output's weight matrix can be optimized by means of supervised training. For each train example x_i and expected output t_i, we compute the output of the hidden layer's neurons, the vector h. Next, we compute the output of the network y and compare it with the expected output t, and adjust each weight in w to reduce the Mean Square Error (MSE) with the Least Mean Squares algorithm (LMS).

This method implies the use of gradient descent (delta rule) and adjusting the weights:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(t_j - y_j)h_i \tag{7}$$

where η is the learning rate ($\eta << 1.0$). This process is repeated for each train example, until the max iteration limit is reached. The center and radius of the neurons is adjusted as well to minimize the output error. We use the error derivative with respect to these parameters, in a fashion similar to that of backpropagation. With $i = 1, ..., m$ hidden neurons, $j = 1, ..., p$ inputs and $k = 1$ output, we update both parameters simultaneously from iteration n to $n + 1$ with:

$$v_{ij}(n+1) = v_i j(n) + \eta_c \frac{[\sum_k (t_k - y_k)w_{ik}]h_i(x_j - v_{ij}(n))}{(\sigma_i(n))^2} \tag{8}$$

$$\sigma_i(n+1) = \sigma_i(n) + \eta_\sigma \frac{[\sum_k (t_k - y_k)w_{ik}]h_i \parallel x - v_i(n) \parallel^2}{(\sigma_i(n))^3} \tag{9}$$

The number of hidden-layer neurons is defined by the user a priori. In our study, we have fixed the number of neurons at 50. The η is set to 0.3. Both η_c and η_σ are set to $\frac{1}{maxIterations}$. The parameter $maxIterations$ denote the maximum number of iterations of the network training algorithm, and is established to 10.

2.3. Learning Vector Quantization

The Learning Vector Quantization (LVQ)[6] family comprises a large spectrum of competitive learning schemes. One of the basic designs that can be used for classification is the LVQ1 algorithm.

An initial set of labeled prototypes is picked by first specifying $n_p \geq c$. Then n_p elements are randomly selected from X to be the initial prototypes, so each class is represented by at least one prototype. LVQ1 has three additional parameters specified by the user: the learning rate $\alpha_k \in (0, 1)$, a constant $\eta \in (0, 1)$, and the terminal number of iterations T. The standard competitive learning update equation is then used to alter the prototype set. If the closed prototype for input x is the vector $v_{i,old}$, the LVQ1 update strategy is

$$v_{i,new} = v_{i,old} + \alpha_k(x - v_{i,old}) \quad when \;\; \ell(v_{i,old}) = \ell(x) \tag{10}$$

or

$$v_{i,new} = v_{i,old} - \alpha_k(x - v_{i,old}) \quad when \;\; \ell(v_{i,old}) \neq \ell(x) \tag{11}$$

Equation 10 rewards the winning prototype for getting the correct label by letting it migrate towards the input vector, while Equation 11 punishes the winning prototype for not labeling the current input correctly by repelling it away from the input vector. In our experiments, the learning rate was updated after each presentation of a new vector using the formula $\alpha_{k+1} = \eta\alpha_k$, $k = 1, \ldots, n-1$; and was restored to the initial, user specified value of α_1 at the end of each pass through X. Before each pass through LVQ1, X is randomly permuted

to avoid dependence of the extracted prototypes on the order of inputs. LVQ1 terminates when either (i) there is no misclassifications in a whole pass through X (and hence, the extracted prototypes are a consistent reference set); or (ii) the prespecified terminal number of iterations is reached.

2.4. Support Vector Machine

A support vector machine (SVM)[10] constructs a hyperplane or set of hyperplanes in a high-dimensional space. A good separation is achieved by the hyperplane that has the largest distance to the nearest training datapoints of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. In order to solve the quadratic programming (QP) problem that arises from SVMs, there are many techniques mostly reliant on heuristics for breaking the problem down into smaller, more-manageable chunks.

The QP problem to train an SVM is shown below:

$$\begin{aligned} max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j k(x_i, x_j) \alpha_i \alpha_j, \\ 0 \leq \alpha_i \leq C, \ \forall i, \\ \sum_{i=1}^{l} y_i \alpha_i = 0. \end{aligned} \tag{12}$$

where $x_i, x_j \in \mathbb{R}^n$.

A point is an optimal point of (12) if and only if the Karush-Kuhn-Tucker (KKT) conditions are fulfilled and $Q_{ij} = y_i y_j k(x_i x_j)$ is positive semi-definite. The KKT conditions are simple; the QP problem is solved when, for all i:

$$\begin{aligned} \alpha_i = 0 \Rightarrow y_i f(x_i) \geq 1, \\ 0 < \alpha_i < C \Rightarrow y_i f(x_i) = 1, \\ \alpha_i = C \Rightarrow y_i f(x_i) \leq 1, \end{aligned} \tag{13}$$

There are several proposal in the literature used to solve this problem. Initial ones like the projected conjugate gradient (PCG) use a "chunking" algorithm based on the fact that the value of the quadratic form is the same if you remove the rows and columns of the matrix that correspond to zero Lagrange multipliers. Therefore, the large QP problem can be broken down into a series of smaller QP problems, whose ultimate goal is to identify all of the non-zero Lagrange multipliers and discard all the Lagrange multipliers. At every step, chunking procedures solves a QP problem that consists of the every non-zero Lagrange multiplier from the last step, and the M worst examples that violate the KKT conditions (13). However, modern procedures suggest a new strategy, avoiding the "chunking" process and speeding up the whole process. A common method for solving the QP problem is the Platt's Sequential Minimal Optimization algorithm considered in this paper[26]. It breaks the problem down into

2-dimensional sub-problems that may be solved analytically, eliminating the need for a numerical optimization algorithm as done in PCG.

In order to solve for the two Lagrange multipliers α_1 and α_2, SMO first computes the constraints on these multipliers and then solves for the constrained maximum. The bound constraints in Equation (12) cause the Lagrange multipliers to lie within a box, while the linear equality constraint in Equation (12) causes the Lagrange multipliers to lie on a diagonal line.

The ends of the diagonal line segment can be expressed quite simply. Without loss of generality, the algorithm first computes the second Lagrange multiplier α_2 and computes the ends of the diagonal line segment in terms of α_2. If the target y_1 does not equal the target y_2, then the following bounds apply to α_2:

$$L = \max(0, \alpha_2^{old} - \alpha_1^{old}), \quad H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}) \tag{14}$$

If the target y_1 equals the target y_2, then the following bounds apply to α_2:

$$L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C), \quad H = \min(C, \alpha_2^{old} + \alpha_1^{old}) \tag{15}$$

The second derivative of the objective function along the diagonal line can be expressed as:

$$\eta = 2k(x_1, x_2) - k(x_1, x_1) - k(x_2, x_2) \tag{16}$$

The next step of SMO is to compute the location of the constrained maximum of the objective function in Equation (12) while allowing only two Lagrange multipliers to change. Under normal circumstances, there will be a maximum along the direction of the linear equality constraint, and η will be less than zero. In this case, SMO computes the maximum along the direction of the constraint.

Therefore SMO will always optimize two Lagrange multipliers at every step, with one of the Lagrange multipliers having previously violated the KKT conditions before the step. That is, SMO will always alter two Lagrange multipliers to move uphill in the objective function projected into the one-dimensional feasible subspace. SMO will also always maintain a feasible Lagrange multiplier vector. Therefore, the overall objective function will increase at every step and the algorithm will converge asymptotically.

3. Data Complexity: Measures of Separability of Classes

In this section we first present a short review on recent studies on data complexity in Subsection 3.1, then we describe the data complexity measures that we have used in this work in Subsection 3.2.

3.1. Recent studies in data complexity

As we have mentioned, data complexity measures are a series of metrics that quantify characteristics which imply some difficulty to the classification task. In the following we gather several recent publications related to these complexity measures and their applications. They can show a picture of the most recent developments in the topic.

- Ho and Basu propose some complexity measures for binary classification problems[17], gathering metrics of three types: overlaps in feature values from different classes; separability of classes; and measures of geometry, topology, and density of manifolds
- Singh offers a review of data complexity measures and proposes two new ones[31].
- Bernadó and Ho investigate the domain of competence of XCS by means of a methodology that characterizes the complexity of a classification problem by a set of geometrical descriptors[5].
- Li et al.[19] analyze some omnivariate decision trees using the measure of complexity based on data density proposed by Ho and Basu.
- Baumgartner and Somorjai define specific measures for regularized linear classifiers[4], using Ho and Basu's measures as reference.
- Sánchez et al. analyze the effect of data complexity on the nearest neighbors classifier[30].
- Dong and Kothari propose[12] a feature selection algorithm based on a complexity measure defined by Ho and Basu.
- Mollineda et al.[24] extend some of Ho and Basu's measure definitions for problems with more than two classes. They analyze these generalized measures in two classic Prototype Selection algorithms and remark that Fisher's discriminant ratio is the most effective for Prototype Selection.
- Sang-Woon and Oommen[18] analyze how to use prototype selection in order to decrease the computation time of several data complexity measures, without severely affecting the outcome with respect to the complete data set.
- García et al.[15] analyze the behavior of the evolutionary prototype selection strategy, considering a complexity measure for classification problems based on overlapping.
- Luengo and Herrera[21] characterized the behavior of the FH-GBML Fuzzy Rule Based Classification System by means of ad-hoc intervals of the data complexity measures[17].

3.2. Measures of Separability of Classes

For our study, we will consider five measures of separability of classes, from the total twelve as described by Ho and Basu[17]. These measures have proven to be the most informative in the classification task[5, 21] due to their high relationship with the class separability and overlapping characteristics. Please note that most of these data complexity measures are only well defined for binary problems.

The measures of separability of classes provide indirect characterizations of class separability. They assume that a class is made up of single or multiple manifolds that form the support of the probability distribution of the given class. The shape, position and interconnectedness of these manifolds give hints on how well two classes are separated, but they do not describe separability by design. The particular descriptions of the measures follows:

L1: *minimized sum of error distance by linear programming.* Linear classifiers can be obtained by a linear programming formulation proposed by Smith[32]. The method minimizes the sum of distances of error points to the separating hyperplane (subtracting a constant margin):

$$\begin{aligned} \text{minimize } & \mathbf{a^t t} \\ \text{subject to } & \mathbf{Z^t w + t \geq b} \\ & \mathbf{t \geq 0} \end{aligned}$$

where $\mathbf{a}$, $\mathbf{b}$ are arbitrary constant vectors (both chosen to be 1), $\mathbf{w}$ is the weight vector to be determined, $\mathbf{t}$ is an error vector, and $\mathbf{Z}$ is a matrix where each column $\mathbf{z}$ is defined on an input vector $\mathbf{x}$ (augmented by adding one dimension with a constant value 1) and its class C (with value C_1 or C_2) as follows:

$$\begin{cases} \mathbf{z} = +\mathbf{x} & \text{if } C = C_1, \\ \mathbf{z} = -\mathbf{x} & \text{if } C = C_2. \end{cases}$$

The value of the objective function in this formulation is used as a measure. The measure has a zero value for a linearly separable problem. We should notice that this measure can be heavily affected by the presence of outliers in the data set.

L2: *error rate of linear classifier by Linear Programming* (LP). This measure is the error rate of the linear classifier defined for L1, measured with the training set. With a small training set this can be a severe underestimate of the true error rate.

N1: *fraction of points on class boundary.* This method constructs a class-blind minimum spanning tree over the entire data set, and counts the number of points incident to an edge going across the two classes. The fraction of such points over all points in the data set is used as a measure.

N2: *ratio of average intra/inter class Nearest Neighbor* (NN) *distance.* For each input instance x_p, we calculate the distance to its nearest neighbor within the class (intraDist(x_p)) and the distance to nearest neighbor of any other class (interDist(x_p)). Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example, i.e.,

$$N2 = \frac{\sum_{i=0}^{m} intraDist(x_i)}{\sum_{i=0}^{m} interDist(x_i)},$$

where m is the number of examples in the data set. This metric compares the within-class spread with the distances to the nearest neighbors of other classes. Low values of this metric suggest that the examples of the same class lie closely

in the feature space. Large values indicate that the examples of the same class are disperse.

N3: *error rate of 1-NN classifier*. This is simply the error rate of a nearest-neighbor classifier measured with the training set. The error rate is estimated by the leave-one-out method. The measure denotes how close the examples of different classes are. Low values of this metric indicate that there is a large gap in the class boundary.

4. Experimental framework

In order to obtain the characterization of the behavior of the ANNs models and the SVM in subsequent sections, their parameters and the problems used as test-bed must be stated. Therefore in this section we first provide details of the data sets used for the experimentation in Subsection 4.1. Next the parameter configuration for the methods studied is presented in Subsection 4.2.

4.1. Data sets generation

We first evaluate the ANNs and the SVM over a set of 438 binary classification problems, in order to obtain their domains of competence. A big quantity of data sets is required because we need to obtain a rich variety of values for each measure of separability of classes. It is not possible to generate data sets with specific values of data complexity measures at this point, so we create many data sets from different original problems in order to fill the value range of each measure.

These first 438 problems are generated from pairwise combinations of the classes of 21 problems from the University of California, Irvine (UCI) repository [2]. These are *iris, wine, new-thyroid, solar-flare, led7digit, zoo, yeast, tae, balanced, car, contraceptive, ecoli, hayes-roth, shuttle, australian, pima, monks, bupa, glass, haberman* and *vehicle*. In order to do that, we construct several new data sets with the combination of the examples from two different classes. This will result in a new data set with only 2 classes and with the original examples which had two such classes as output. We perform this process for every possible pairwise combination of classes, and we compute the five measures of separability of classes for every data set. If an obtained data set with this procedure proves to be linearly-separable, we discard it. The complexity measure L1 indicates if a problem is linearly-separable when its value is equal to zero.

This method for generating binary data sets is limited by the proper combinatorics, and we can only obtain over 200 new data sets with the original 21 data sets with this first approach. In order to obtain more data sets, we group the classes two by two, that is, we create a new binary data set, and each of its two classes are the combination of two original classes each. For this second approach we have used *ecoli*, *glass* and *flare* data sets, since they have a high number of class labels, obtaining 238 new ones. Again, those data sets with a L1 value of zero are discarded.

A second bunch of data sets will be considered in order to validate the results obtained in our analysis. We have applied the same methodology of grouping

the classes two by two to the *yeast, flare* and *car* data sets. With these last three data sets we have obtained another 200 non-linearly separable data sets used for validating the rules obtained in our analysis, for a total of 638 data sets.

In order to measure the ANNs and the SVM accuracy performance in each data set, we have applied a 10-fcv validation scheme.

4.2. Parameter settings

The parameter selection for ANN and SVM models is a key question in order to obtain a good performance and avoid over-learning. The common process implies to train the model with different topologies and learning parameters over the problems (i.e. using a parameter grid) and to chose the best configuration.

When analyzing the performance of these models with respect to the measures of separability of classes we consider to use the same configuration for all the data sets in order to relate results across all the data sets. Different parameter configurations of the models may produce very different outcomes, and we want to relate the good or bad outputs to the measures of separability of classes values as independently of external factors as possible. Moreover, a particular configuration of the models for each data set built as indicated in the previous section would be very time consuming.

Therefore, we have selected the same parameters for all the 438 initial data sets in each model, trying to obtain a good overall performance. These parameters have been chosen from a range possibilities, finally using those which offer the best average performance all over the initial 438 data sets. These ranges follows:

- MLP, RBFN and LVQ number of neurons: from 10 to 50, using increments of 10.
- SVM C parameter: from 1 to 100, using increments of 10.
- MLP, RBFN, LVQ and SVM learning parameters (α, η, ω, σ): from 0.1 to 1, using increments of 0.1.

The final best overall parameter configuration as required by KEEL software for each model is summarized in Table 1.

Table 1: Parameters used for the models

MLP	Hidden Layers = 1, Neurons per Layer = 40
RBFN	Neurons = 50
LVQ	$n_p = 40$, $\alpha = 0.3$, $\eta = 0.8$
SVM	$C = 1$, tolerance = 0.001, kernel = Puk, $\epsilon = 1^{-12}$, $\omega = 1$, $\sigma = 1$

In Table 2 we have summarized the global Training and Test accuracy for the initial 438 data sets obtained by the ANN methods and the SVM with these parameter configuration.

Table 2: Global Average Training and Test Accuracy/std. dev. for RBFN, MLP, LVQ and SVM

	Global % Accuracy Training Global Training std. dev.	Global % Accuracy Test Global Test std. dev.
RBFN	93.12% 7.99	90.65% 10.42
MLP	96.82% 4.40	87.02% 11.68
LVQ	86.44% 11.93	86.35% 12.46
SVM	94.72% 6.07	91.99% 9.13

5. Analysis of the Learning Methods with Measures of Separability of Classes

In this study, our aim is to analyze the characterization of the behavior of the for learning methods by means of the data complexity measures. In order to do this analysis, we divide this section into the following subsections. First we introduce the motivation for the use of the measures of separability of classes in order to characterize the models' performance in Subsection 5.1. In Subsection 5.2 we describe the process used in order to discriminate the informative measures of separability of classes. Next we determine several rules based on intervals of the selected data complexity measures for the ANNs and SVM in Subsection 5.3. Then we determine the domains of competence of the four models from the single rules in Subsection 5.4 and validate these domains in Subsection 5.5.

5.1. Motivation

One first indicator of the performance of the ANN or SVM considered could be the average accuracy obtained in the training partitions. However, it is well known that it is possible to increase such accuracy in detriment of the test accuracy. This phenomena is usually known as over-learning in the classification topic, and it discourages the use of the training accuracy as a precise indicator of the true performance of the model.

Figures 1 to 4 contains the results of the four models showing the training and test accuracy over all the initial 438 data sets, plotted in ascending training accuracy value for the RBFN, MLP, LVQ and SVM models respectively. The dark line in these figures represent the training values, and the variability in test values is patent. We want to point out that we can find over-learning in all four Figures 1 to 4, especially in the highest training values.

An alternative option is to use the measures of separability of classes to analyze when the model produces a good or bad outcome for the data set. Since these metrics measure aspects on the class separability of the data set, the good or bad outcome of the method for a given configuration can be stated independently of the learning method used afterwards.

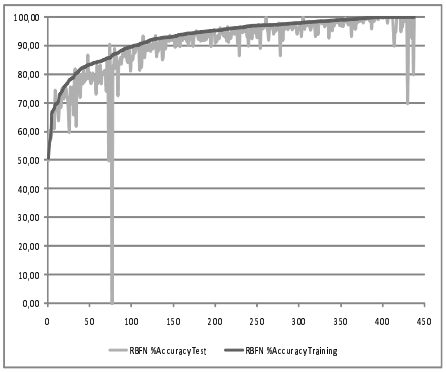

Figure 1: Accuracy in Training/Test for RBFN sorted by training accuracy

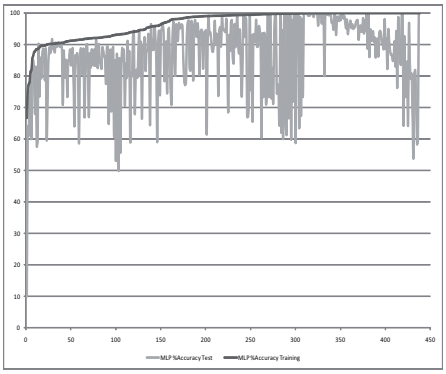

Figure 2: Accuracy in Training/Test for MLP sorted by training accuracy

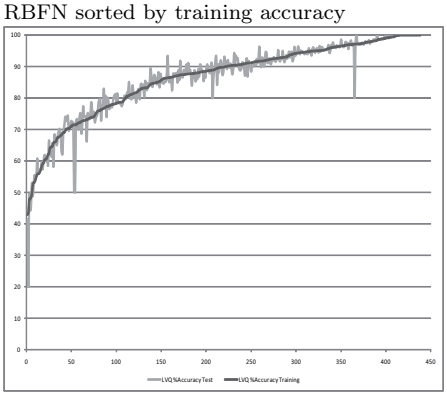

Figure 3: Accuracy in Training/Test for LVQ sorted by training accuracy

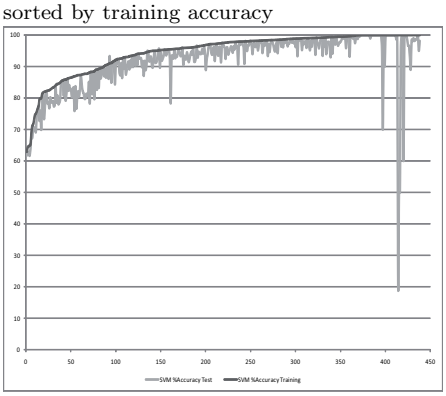

Figure 4: Accuracy in Training/Test for SVM sorted by training accuracy

5.2. Selection of the informative measures of separability of classes

Given the initial 438 data sets described in Subsection 4.1, we will obtain 438 values of each measure of separability of classes. We try to extract regions of these measures in which all the four models behave well or badly. In order to do so, for each complexity measure the data-sets are sorted by the ascending value of such complexity measure, and put altogether in a figure. In the X axis the data sets are depicted with the same distance between them. The reason to do so is to avoid huge gaps between metric values to distort the interval extraction procedure described next. The Y axis depicts the accuracy obtained both in training and test for the model.

Once the data set have been arranged as described above, we have extracted different *ad-hoc* intervals which present *good* or *bad behavior* for the ANNs and the SVM *simultaneously*. The considered criteria for this analysis follows as a representative example of good and bad behavior in classification. Nevertheless it must be stated that it can be adjusted to each one's necessities.

- We understand for *good behavior* an average high test accuracy in the interval, at least 90%.

- By *bad behavior* we refer to the presence of over-fitting and/or average low test accuracy in the interval (under 80%).

These intervals must also contain a significative number of data sets in order to avoid biases introduced by small good/bad intervals located in bad/good regions respectively. On the other hand, if only a few data sets do not verify these indications, the interval can be extracted as well, as these data sets would act as outliers in this case.

Not always is possible to extract shared intervals in the figures for all the four models with the mentioned characteristics. For example, in Figure 5 we show an example of a figure without any quality interval.

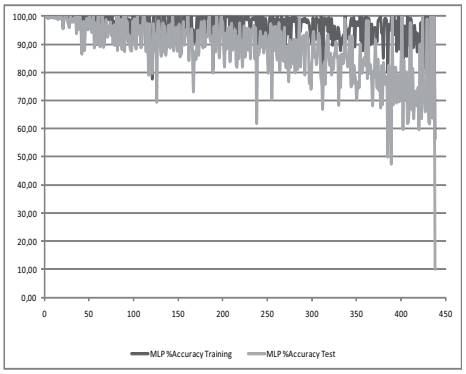

Figure 5: MLP accuracy in Training/Test sorted by L2

For some data complexity measures, there are one or more intervals of either good or bad behavior for all the four learning methods. In particular for the N1 and N3 measures of separability of classes we can extract common intervals of good and bad behavior for the four learning methods. These two measures are specially designed to estimate both the proximity of examples of different classes in their boundaries and to indicate wide gaps between such boundaries.

In Figures 6 to 13 we present the figures for the N1 and N3 measures in which we have extracted the *ad-hoc* intervals which present good or bad behavior for all the ANNs and SVM with the same size, using a vertical line to delimit them.

5.3. Determination of Rules Based on the Learning Methods' Behavior

In Table 3 we summarize the particular boundary values of the ad-hoc intervals depicted in Figures 6 to 13 for the N1 and N3 complexity measures.

From these ad-hoc intervals is possible to construct several rules that model the performance of the ANNs and the SVM. In Table 4 we have summarized the rules derived from Table 3 using the intervals as the antecedents of the rules. Given a particular data set X, we get the complexity measure of X with the notation $CM[X]$. Table 4 is organized with the following columns.

- The first column corresponds to the identifier of the rule for further references.

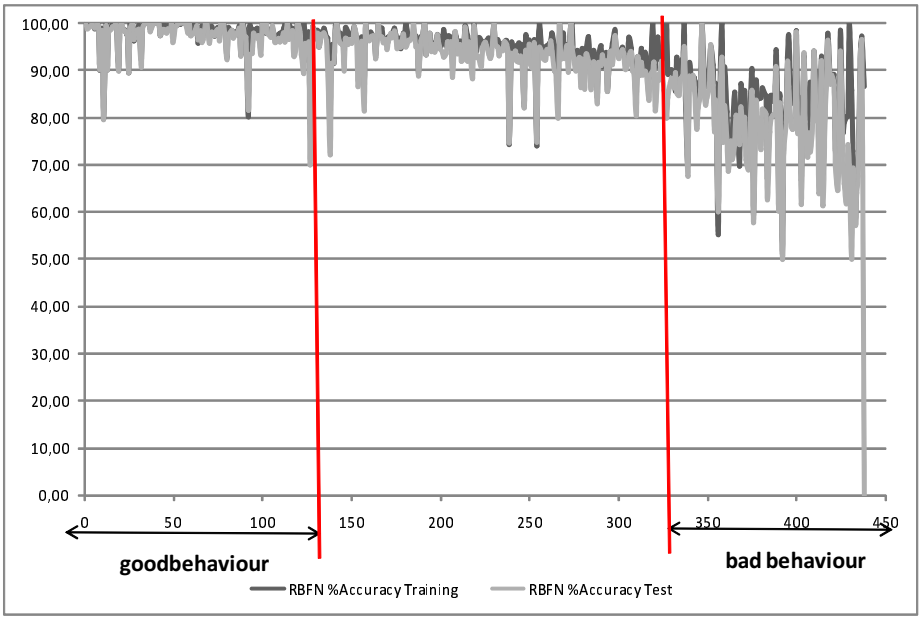

Figure 6: RBFN accuracy in Training/Test sorted by N1

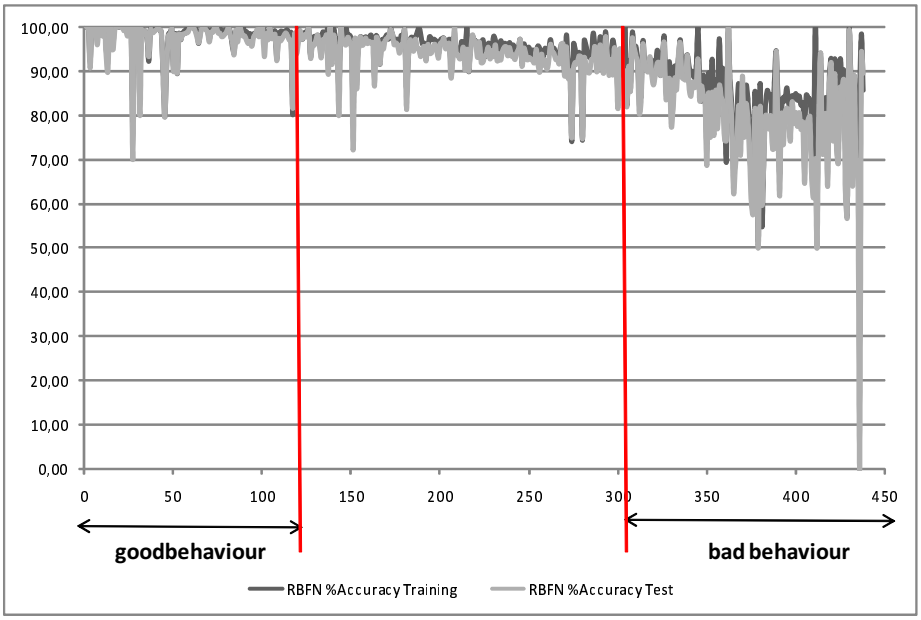

Figure 7: RBFN accuracy in Training/Test sorted by N3

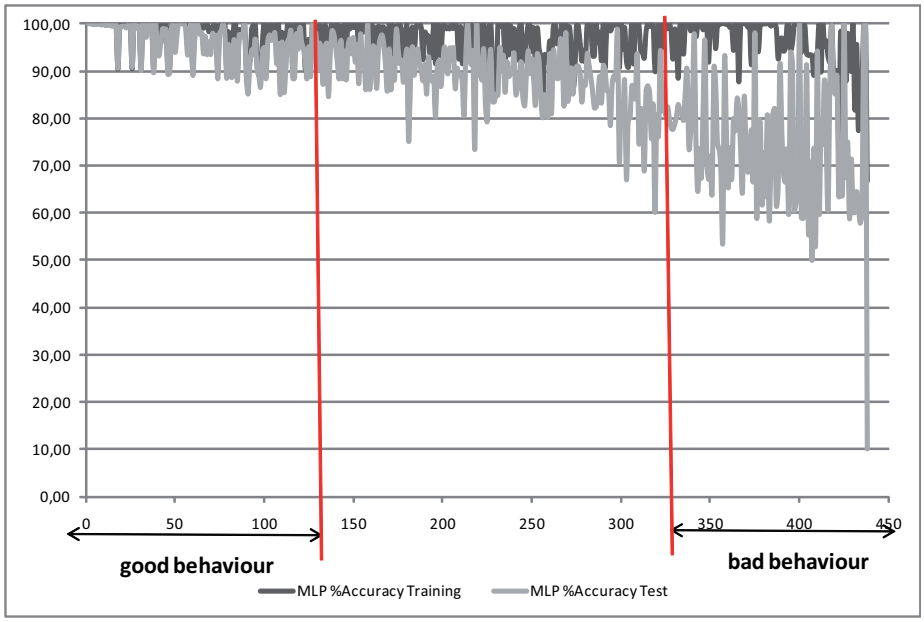

Figure 8: MLP accuracy in Training/Test sorted by N1

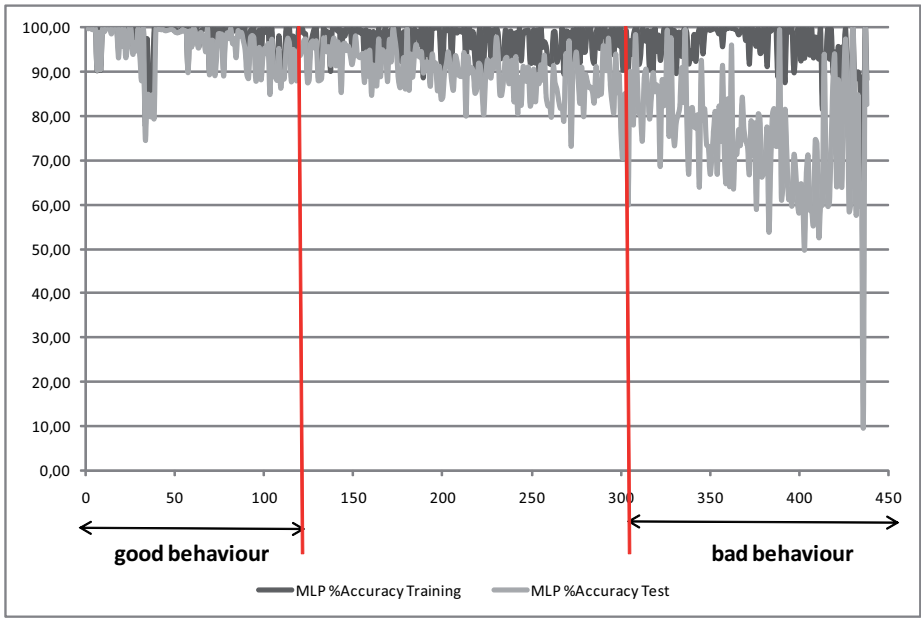

Figure 9: MLP accuracy in Training/Test sorted by N3

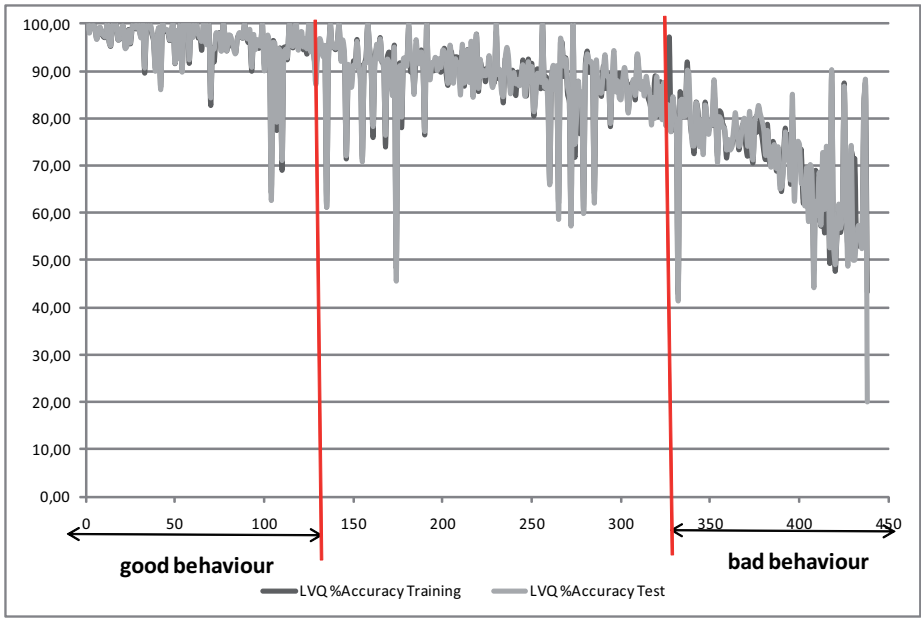

Figure 10: LVQ accuracy in Training/Test sorted by N1

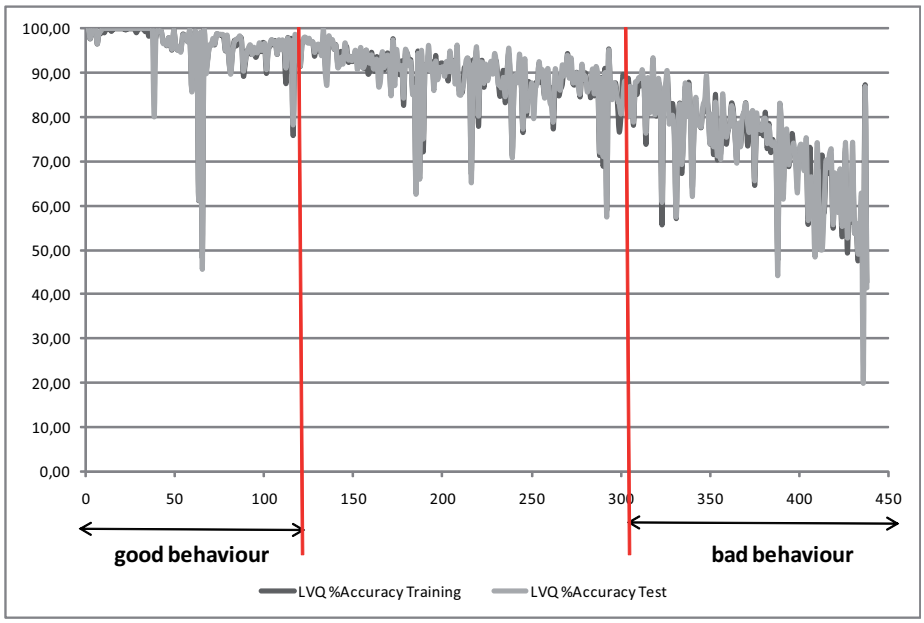

Figure 11: LVQ accuracy in Training/Test sorted by N3

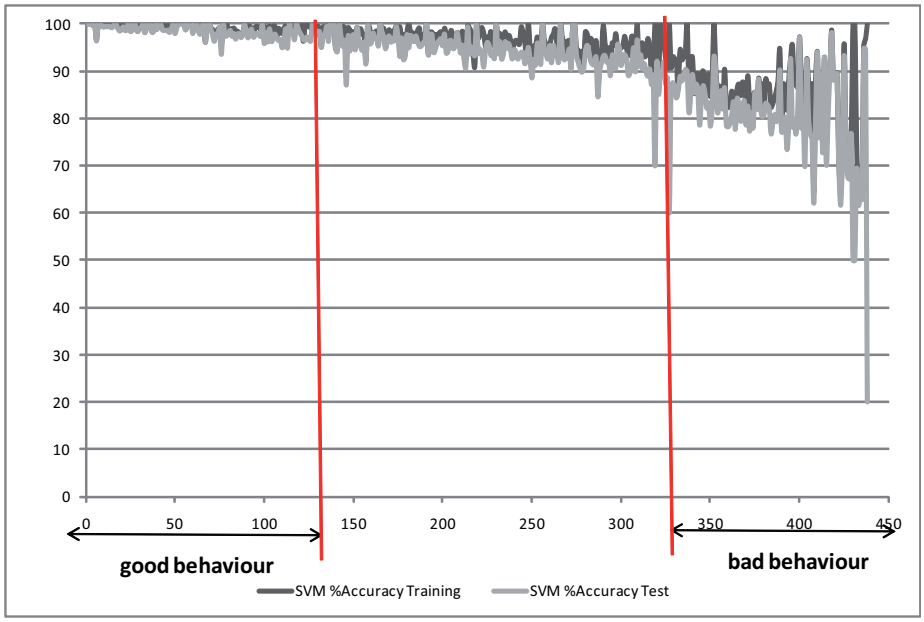

Figure 12: SVM accuracy in Training/Test sorted by N1

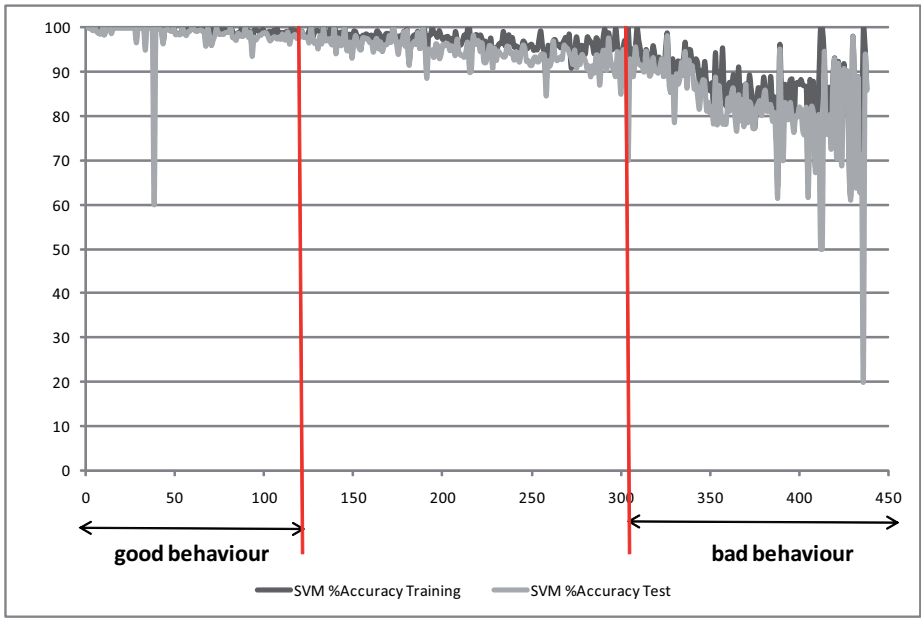

Figure 13: SVM accuracy in Training/Test sorted by N3

Table 3: Significant intervals

Interval	ANNs behavior
$N1 < 0.08$	*good behavior*
$N3 < 0.029$	*good behavior*
$N1 > 0.25$	*bad behavior*
$N3 > 0.108$	*bad behavior*

- The "Rule‘" column presents the rule itself.
- The third column "Support" presents the percentage of data sets which verifies the antecedent of the rule.
- The column "Model‘ identifies the classification model to which this row refers to.
- The column "% Training" shows the average accuracy in training of all the examples which are covered by the rule.
- The column "Training Diff." contains the difference between the training accuracy of the rule and the training accuracy across all 438 data sets presented in Table 2.
- The column "% Test" shows the average accuracy in test of all the examples which are covered by the rule.
- The column "Test Diff." contains the difference between the test accuracy of the rule and the test accuracy across all 438 data sets presented in Table 2.

The positive rules (denoted with a "+" symbol in their identifier) always show a positive difference with the global average, both in training and test accuracy. The negative ones (with a "-" symbol in their identifier) verify the opposite case. The support of the rules shows us that we can characterize a wide range of data sets and obtain significant differences in accuracy.

From this set of rules we can state the following:

- A low value of N1 results in a good behavior of all the learning models considered. That means that those problems with a low percentage of instances of different classes that are closer than to examples of their own class are adequate for the ANNs and SVM. They are characterized by a good separation of examples in the class boundaries.
- A low value of N3 results in a good behavior of all the learning models considered. When a wide gap between the class boundary is present, it is easy to shape the separation between the classes and to obtain a good model for the problem.

Table 4: Rules with one metric obtained from the intervals

Id.	Rule	Support	Model	%Training	Training Diff.	% Test	Test Diff.
R1+	If N1[X] < 0.08 then good behavior	29.22%	RBFN	98.03%	4.91%	97.20%	6.55%
			MLP	97.72%	1.69%	95.95%	7.56%
			LVQ	95.65%	10.26%	95.86%	10.47%
			SVM	98.45%	6.91%	97.93%	8.15%
R2+	If N3[X] < 0.029 then good behavior	26.71%	RBFN	98.18%	5.06%	97.28%	6.63%
			MLP	97.31%	1.33%	96.02%	7.63%
			LVQ	95.67%	10.28%	95.83%	10.44%
			SVM	98.66%	7.12%	98.09%	8.31%
R1-	If N1[X] > 0.25 then bad behavior	24.43%	RBFN	83.64%	-9.48%	78.22%	-12.43%
			MLP	93.68%	-2.30%	76.74%	-11.65%
			LVQ	70.53%	-14.86%	70.02%	-15.37%
			SVM	80.38%	-11.16%	76.81%	-12.97%
R2-	If N3[X] > 0.108 then bad behavior	31.51%	RBFN	85.33%	-7.79%	80.64%	-10.01%
			MLP	93.80%	-2.18%	78.47%	-9.92%
			LVQ	72.87%	-12.52%	72.63%	-12.76%
			SVM	82.34%	-9.20%	79.03%	-10.75%

- A high value of N1 results in a bad behavior of all the learning models considered. Those problems with a high percentage of instances being close to other class instances are difficult to model by the considered learning methods.

- A high value of N3 results in a bad behavior of all the learning models considered. When the boundaries of the two classes are getting closer, the performance of the learning model decreases significatively.

With this information is possible to know in advance when the problem will be difficult for the ANN o the SVM model calculating the N1 and N3 measures before building the model. Therefore, it could be possible to relate the *good* or *bad behavior* consequents of the rules with the necessity of a more sophisticated topology of the net, or with the use of more complex kernels for the SVM. The contrary case would be also verified, saving time in the model configuration and construction.

Although we have obtained some interesting individual rules, we can extend our study by considering the combination of these complexity metrics in order to obtain more interpretable, extensive and descriptive rules.

5.4. Collective Evaluation of the Set of Rules

The objective of this section is to analyze the good and bad rules jointly. We have considered the disjunctive combination (we use the *or* operator) of all the positive rules to obtain a single rule (Positive Rule Disjunction -PRD-). The same procedure is performed with all the negative ones so we obtain another rule (Negative Rule Disjunction -NRD-). The new disjunctive rules will be activated if any of the component rules' antecedents are verified.

By means of merging the individual rules we can arrive at a more general description, with a wider support, of the behavior of the learning models. In Table 5 we summarize both disjunctions, and a third rule representing those data sets which are not characterized by either one.

Table 5: Disjunctive rules obtained from single rules

Id.	Rule	Support	Model	%Training	Training Diff.	% Test	Test Diff.
PRD	If R1+ or R2+ then good behaviour	32.65%	RBFN	98.16%	5.04%	97.11%	6.46%
			MLP	97.29%	0.47%	95.17%	8.15%
			LVQ	95.52%	9.08%	95.58%	9.23%
			SMO	99.27%	7.00%	98.30%	8.12%
NRD	If R1- or R2- then bad behaviour	31.96%	RBFN	85.50%	-7.62%	80.81%	-9.84%
			MLP	96.12%	-0.70%	75.68%	-11.34%
			LVQ	74.04%	-12.40%	73.73%	-12.62%
			SMO	87.67%	-8.99%	82.68%	-10.59%
not characterized	If not PRD and not NRD then good behaviour	35.39%	RBFN	95.35%	2.23%	93.59%	2.94%
			MLP	97.03%	0.21%	89.74%	2.72%
			LVQ	89.26%	2.82%	89.24%	2.89%
			SMO	96.89%	1.64%	94.57%	2.08%

From the collective rules we can observe that the support has been increased from the single rules for PRD. NRD obtains similar support with respect to the single rules which conform it. It is also important to notice that the Test and Training Accuracy Differences are maintained with respect to the single rules from Table 4 despite the increment in support.

Since no data sets are shared by PRD and NRD, we can consider three blocks of data sets with their respective support, as depicted in Figures 14 to 17 (with no particular data set order within each block):

- The first block (the left-side one) represents the data sets covered by the PRD rule. They are the data sets recognized as being those in which the ANNs and the SVM have good behavior.
- The second block (the middle one) plots the data sets for the NRD rule, which are bad data sets for the methods considered.
- The third and last block (the right-side one) contains the unclassified data sets by the previous two rules.

We can see that almost the 65% of the analyzed data sets are covered by these two rules with significant differences. Hence the PRD and NRD rules can be used to define the shared domains of competence of the ANNs and SVM models.

5.5. Domains of competence validation

In this section the domains of competence obtained by means of the PRD and NRD rules are validated by means of a fresh bunch of 200 data sets. In Table 6 we have summarized the average training and test accuracy values for these new data sets (using the same parameters as before).

Table 7 presents the PRD and NRD rules average training and test accuracy for the validation data sets for each learning model, as well as the difference from the global average depicted in Table 6.

The three blocks figure representation of the data of Table 7 for the MLP, RBFN, LVQ and SVM methods are depicted in Figures 18, 19, 20 and 21 respectively.

From these results we can observe the following:

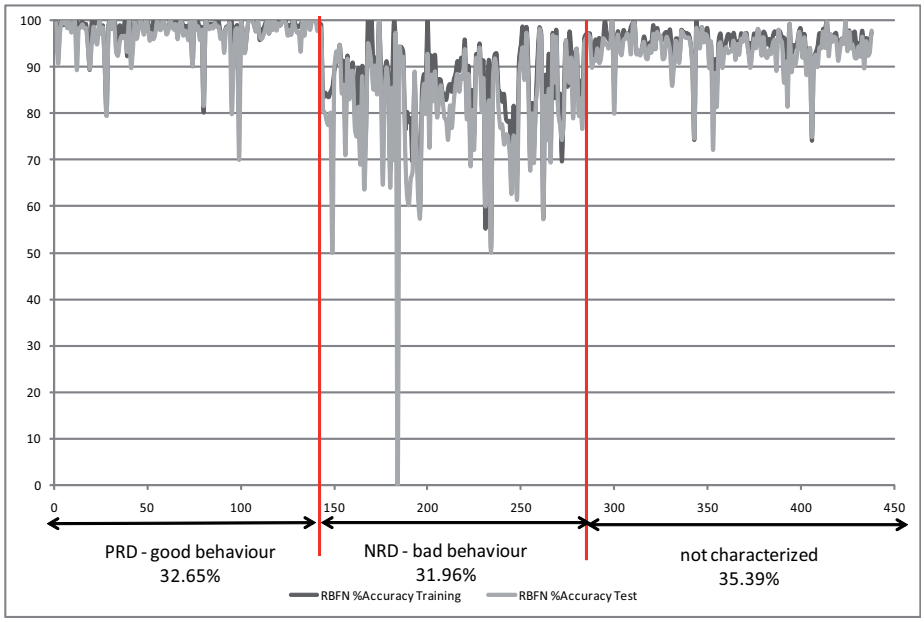

Figure 14: Three blocks representation for PRD, NRD and not covered data sets for RBFN

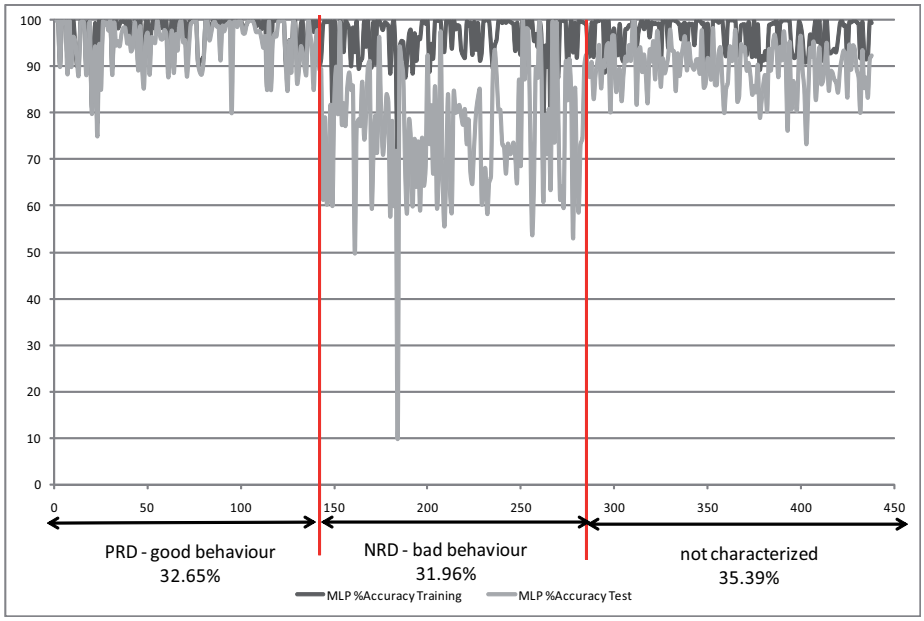

Figure 15: Three blocks representation for PRD, NRD and not covered data sets for MLP

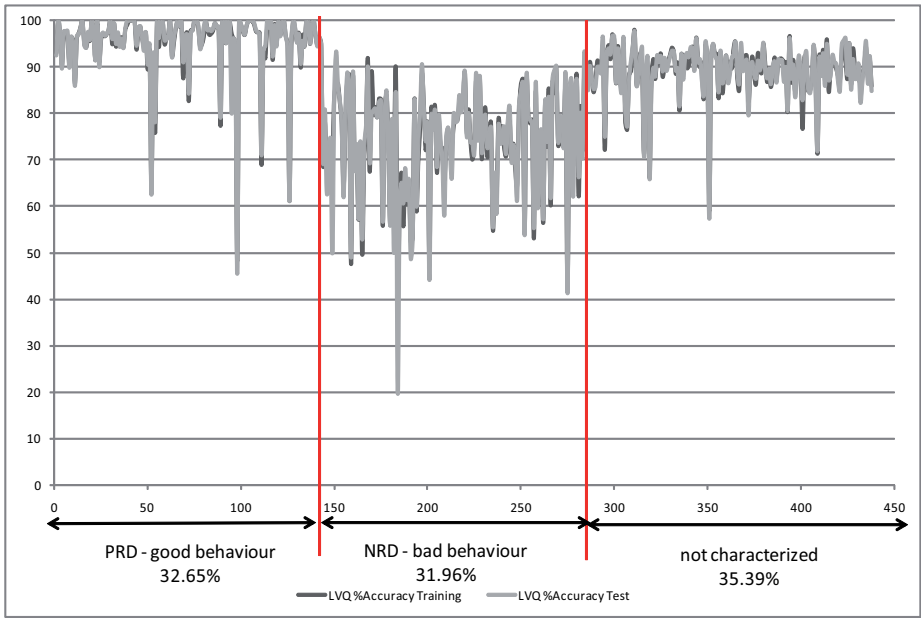

Figure 16: Three blocks representation for PRD, NRD and not covered data sets for LVQ

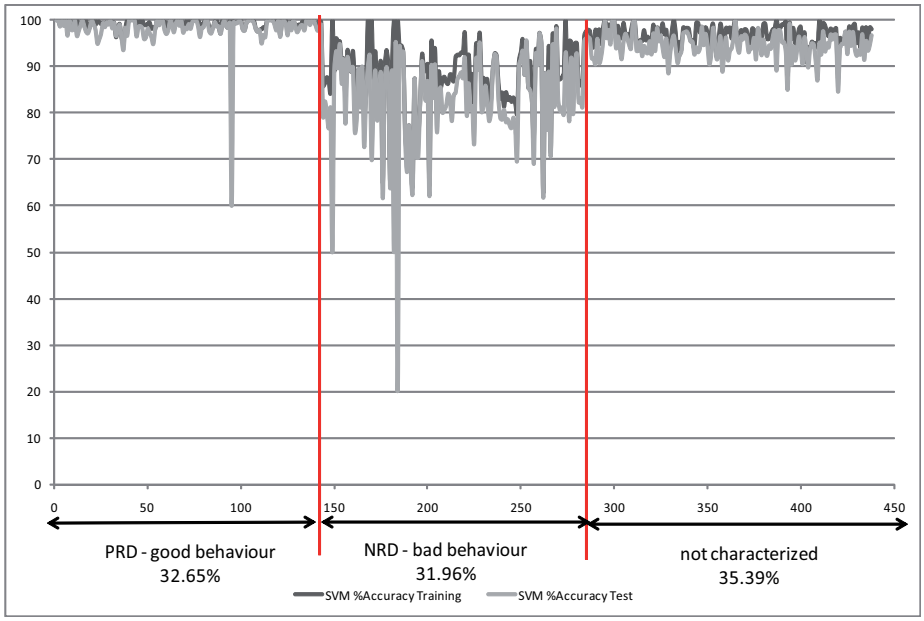

Figure 17: Three blocks representation for PRD, NRD and not covered data sets for SVM

Table 6: Global Average Training and Test Accuracy/std. dev. for RBFN, MLP, LVQ and SVM over the validation data sets

	Global % Accuracy Training Global Training std. dev.	Global % Accuracy Test Global Test std. dev.
RBFN	85.23% 11.51	84.84% 11.57
MLP	93.09% 6.90	88.89% 9.11
LVQ	83.12% 13.56	83.28% 13.63
SVM	91.57% 9.11	90.35% 9.77

Table 7: Validation results for PRD and NRD rules

Id.	Rule	Support	Model	%Training	Training Diff.	% Test	Test Diff.
PRD	If R1+ or R2+ then good behaviour	34.50%	RBFN	92.88%	7.65%	92.82%	7.98%
			MLP	96.50%	3.41%	95.33%	6.44%
			LVQ	93.63%	10.51%	93.71%	10.43%
			SMO	97.96%	7.00%	97.70%	8.12%
NRD	If R1- or R2- then bad behaviour	37.50%	RBFN	76.29%	-8.94%	75.48%	-9.36%
			MLP	88.01%	-5.08%	80.28%	-8.61%
			LVQ	69.79%	-13.33%	69.98%	-13.30%
			SMO	83.38%	-8.99%	80.82%	-10.59%
not characterized	If not PRD and not NRD then good behaviour	28.00%	RBFN	87.78%	2.55%	87.53%	2.69%
			MLP	95.71%	2.62%	92.46%	3.57%
			LVQ	88.02%	4.90%	88.23%	4.95%
			SMO	94.67%	1.64%	94.04%	2.08%

- The bad behavior characterized by NRD is maintained in this new set of data sets. The data sets covered by this rule show a bad performance for the learning methods considered.
- The PRD rule covers the new data sets for which the four learning methods perform remarkably well. Only in the case of LVQ appear some isolated outliers with test values below the 70%.
- The differences for these two rules for the four models in training and test are similar to those obtained in the previous section when extracting the characterization.
- The support of the PRD and NRD rules are similar to the "training" case. The support of the not characterized region is also similar.
- The test accuracy differences of the not characterized data sets are positive, but clearly below than the obtained by the PRD rule.

From these statements, it is possible to conclude that the obtained domains of competence and their interpretation can indicate the characteristics of the data sets for which the ANNs and the SVM models considered would perform well or badly.

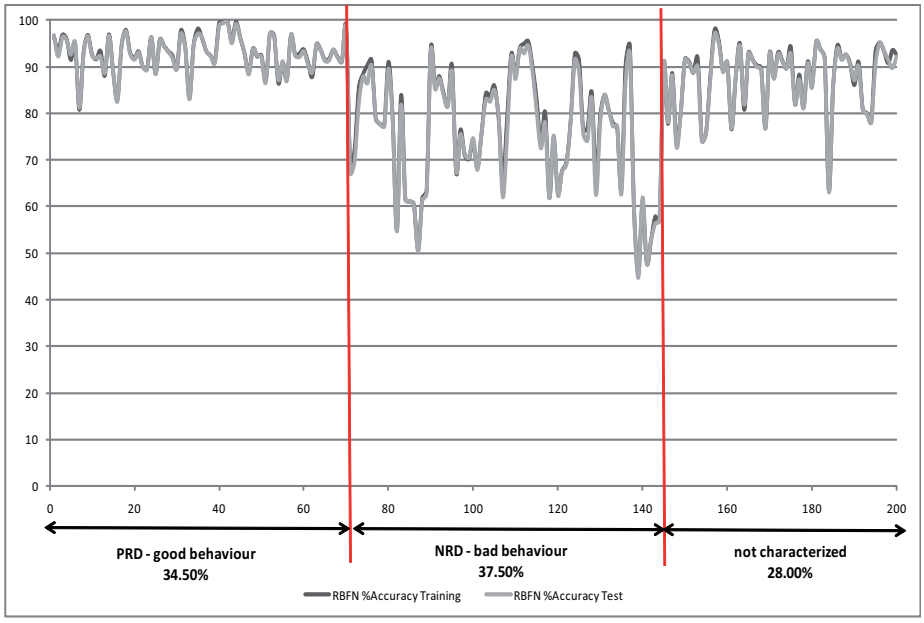

Figure 18: Three blocks representation for PRD, NRD and not covered data sets for RBFN considering validation data sets

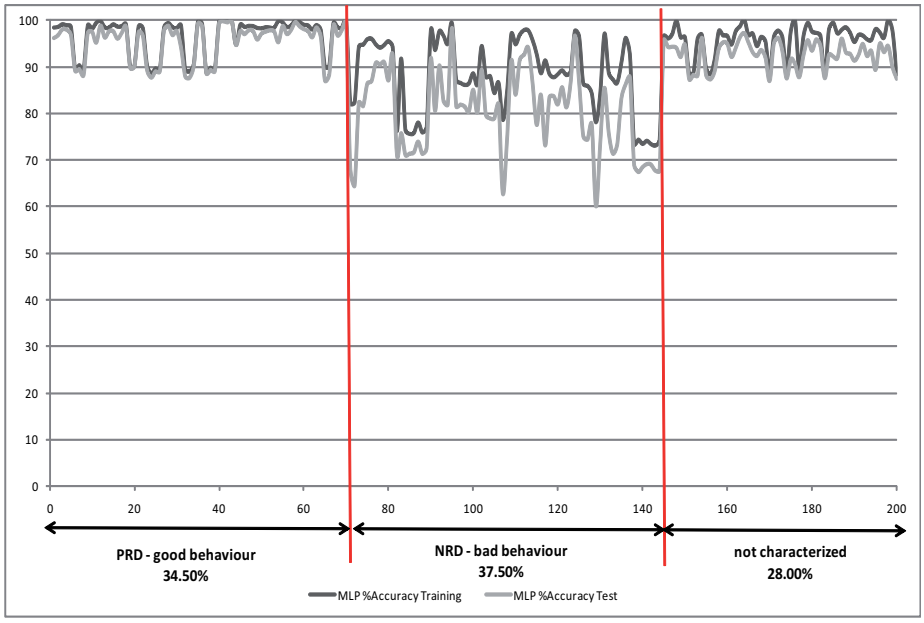

Figure 19: Three blocks representation for PRD, NRD and not covered data sets for MLP considering validation data sets

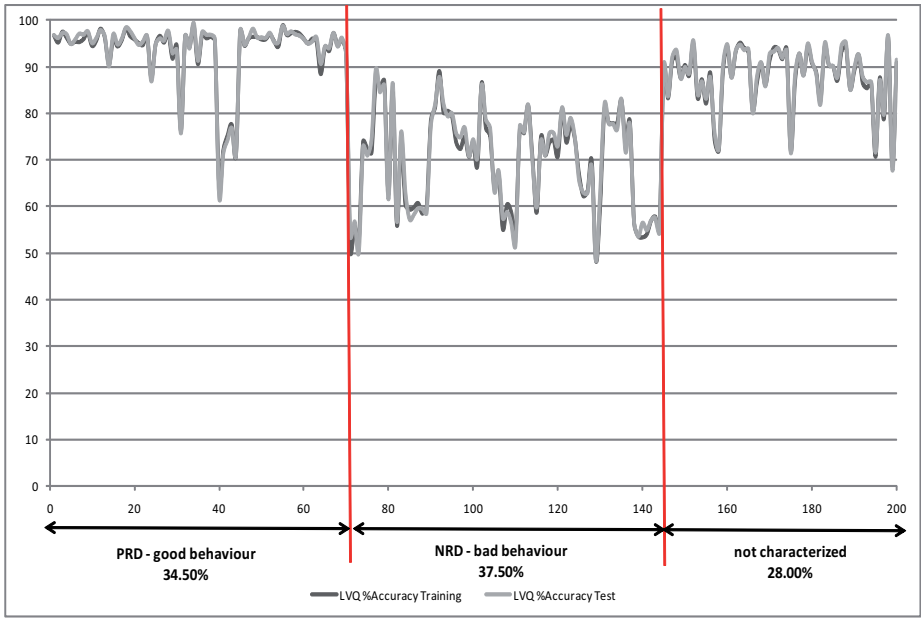

Figure 20: Three blocks representation for PRD, NRD and not covered data sets for LVQ considering validation data sets

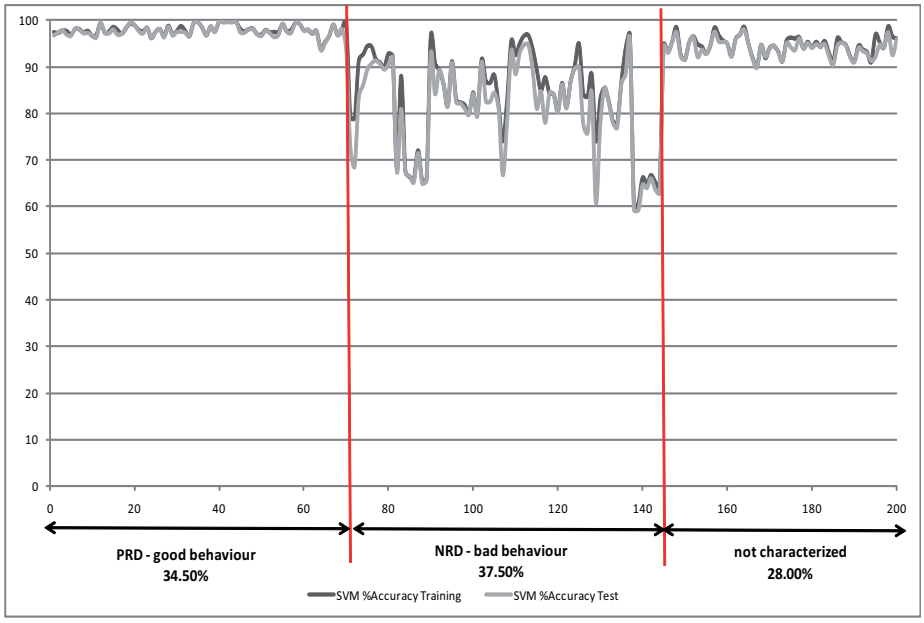

Figure 21: Three blocks representation for PRD, NRD and not covered data sets for SVM considering validation data sets

6. Concluding Remarks

We have performed a study over a set of binary data sets with three ANN methods and one SVM model. We have computed five data complexity measures for the data sets known as *measures of separability of classes* in order to obtain intervals of such metrics in which the method's performance is significantly good or bad. We have obtained descriptive rules for two measures, and studied the interaction between them.

We have obtained two final rules which are simple, interpretable and precise to describe the common good and bad performance of the ANNs and the SVM models considered in this work, thus establishing their shared domains of competence. These domains of competence have been validated using an extra amount of data sets, observing that they generalize well and describe the behavior of the four models appropriately. Furthermore, we present the possibility of determining for which data sets RBFN, MLP LVQ and SVM will perform well or badly prior to their execution using the obtained domains of competence.

We must point out that this is a particular study for four specific methods. On the other hand, this work presents a new challenge that could be extended to other ANN models or SVM methods, to analyze the relation between their domains of competence and parameter adjustment, and to develop new measures which could give more information on the behaviors of ANNs and SVMs for pattern recognition.

References

[1] M. Asaduzzaman, M. Shahjahan, K. Murase, Faster training using fusion of activation functions for feed forward neural networks, International Journal of Neural Systems 19 (2009) 437–448.

[2] A. Asuncion, D. Newman, UCI machine learning repository, 2007.

[3] M. Basu, T.K. Ho, Data Complexity in Pattern Recognition (Advanced Information and Knowledge Processing), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] R. Baumgartner, R.L. Somorjai, Data complexity assessment in undersampled classification of high-dimensional biomedical data, Pattern Recognition Letters 12 (2006) 1383–1389.

[5] E. Bernadó-Mansilla, T.K. Ho, Domain of competence of XCS classifier system in complexity measurement space, IEEE Transactions on Evolutionary Computation 9 (2005) 82–104.

[6] J.C. Bezdek, L. Kuncheva, Nearest prototype classifier designs: An experimental study, International Journal of Intelligent Systems 16 (2001) 1445–1473.

[7] D. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.

[8] M. Buhmann, Radial Basis Functions: Theory and Implementations, Cambridge Monographs on Applied and Computational Mathematics, 2003.

[9] R. Capparuccia, R. De Leone, E. Marchitto, Integrating support vector machines and neural networks, Neural Networks 20 (2007) 590–597.

[10] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.

[11] G. Daqi, L. Chunxia, Y. Yunfan, Task decomposition and modular single-hidden-layer perceptron classifiers for multi-class learning problems, Pattern Recognition 40 (2007) 2226–2236.

[12] M. Dong, R. Kothari, Feature subset selection using a new definition of classificabilty, Pattern Recognition Letters 24 (2003) 1215–1225.

[13] D. Du, K. Li, M. Fei, A fast multi-output rbf neural network construction method, Neurocomputing 73 (2010) 2196–2202.

[14] D. Fisch, B. Kühbeck, B. Sick, S.J. Ovaska, So near and yet so far: New insight into properties of some well-known classifier paradigms, Information Sciences 180 (2010) 3381–3401.

[15] S. García, J.R. Cano, E. Bernadó-Mansilla, F. Herrera, Diagnose of effective evolutionary prototype selection using an overlapping measure, International Journal of Pattern Recognition and Artificial Intelligence 23 (2009) 2378–2398.

[16] A. Ghosh, M. Biehl, B. Hammer, Performance analysis of lvq algorithms: a statistical physics approach, Neural Networks 19 (2006) 817–829.

[17] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 289–300.

[18] S.W. Kim, B.J. Oommen, On using prototype reduction schemes to enhance the computation of volume-based inter-class overlap measures, Pattern Recognition 42 (2009) 2695–2704.

[19] Y. Li, M. Dong, R. Kothari, Classifiability-based omnivariate decision trees, IEEE Transactions on Neural Networks 16 (2005) 1547–1560.

[20] J. Luengo, S. García, F. Herrera, A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests, Expert Systems with Applications 36 (2009) 7798–7808.

[21] J. Luengo, F. Herrera, Domains of competence of fuzzy rule based classification systems with data complexity measures: A case of study using a fuzzy hybrid genetic based machine learning method, Fuzzy Sets and Systems 161 (2010) 3–19.

[22] S. Maldonado, R. Weber, J. Basak, Simultaneous feature selection and classification using kernel-penalized support vector machines, Information Sciences 181 (2010) 115–128.

[23] M.F. Moller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.

[24] R.A. Mollineda, J.S. Sánchez, J.M. Sotoca, Data characterization for effective prototype selection, in: Proc. of the 2nd Iberian Conf. on Pattern Recognition and Image Analysis, Springer, 2005, pp. 27–34.

[25] Y.J. Oyang, S.C. Hwang, Y.Y. Ou, C.Y. Chen, Z.W. Chen, Data classification with radial basis function networks based on a novel kernel density estimation algorithm, IEEE Transactions on Neural Networks 16 (2005) 225–236.

[26] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in kernel methods: support vector learning, MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.

[27] M.J. Powell, Algorithm for approximation, Algorithm for approximation, Clarendon Press, Oxford, 1987, pp. 143–168.

[28] C. Renjifo, D. Barsic, C. Carmen, K. Norman, G.S. Peacock, Improving radial basis function kernel classification through incremental learning and automatic parameter selection, Neurocomputing 72 (2008) 3–14.

[29] R. Rojas, Neural Networks: A Systematic Introduction, Springer, 1996.

[30] J.S. Sánchez, R.A. Mollineda, J.M. Sotoca, An analysis of how training data complexity affects the nearest neighbor classifiers, Pattern Analysis & Applications 10 (2007) 189–201.

[31] S. Singh, Multiresolution estimates of classification complexity, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 1534–1539.

[32] F.W. Smith, Pattern classifier design by linear programming, IEEE Transactions on Computers 17 (1968) 367–372.

[33] S. Suresh, N. Sundararajan, P. Saratchandran, Risk-sensitive loss functions for sparse multi-category classification problems, Information Sciences 178 (2008) 2621–2638.

[34] Q. Tao, D. Chu, J. Wang, Recursive support vector machines for dimensionality reduction, IEEE Transactions on Neural Networks 19 (2008) 189–193.

2.3. An Automatic Extraction Method of the Domains of Competence of Fuzzy Rule Based Classification Systems using Data Complexity Measures

- J. Luengo, F. Herrera, An Automatic Extraction Method of the Domains of Competence of Fuzzy Rule Based Classification Systems using Data Complexity Measures. **Submitted to IEEE Transactions on Fuzzy Systems**.
  - Status: **Submitted**.
  - Impact Factor (JCR 2009): 3.291.
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 6 / 103.
  - Subject Category: Engineering, Electrical & Electronic. Ranking 10 / 246.

An Automatic Extraction Method of the Domains of Competence of Fuzzy Rule Based Classification Systems using Data Complexity Measures

Journal:	*Transactions on Fuzzy Systems*
Manuscript ID:	TFS-2010-0520
Manuscript Type:	Full Papers
Keywords:	Classification, Data Complexity, Fuzzy Rule Based Systems, Domains of Competence

An Automatic Extraction Method of the Domains of Competence of Fuzzy Rule Based Classification Systems using Data Complexity Measures

Julián Luengo and Francisco Herrera

Abstract

When dealing with problems using Fuzzy Rule Based Classification Systems it is difficult to know in advance whether the model will perform well or poorly. It would be useful to have a procedure which indicates, prior to the application of the Fuzzy Rule Based Classification System, if the outcome will be good or bad.

In this work we present an automatic extraction method to determine the domains of competence of Fuzzy Rule Based Classification Systems by means of data complexity measures. It uses twelve metrics of data complexity acting over a large benchmark of data sets in order to analyze the behavior patterns of the method, obtaining intervals of data complexity measures with good or bad performance. The Fuzzy Hybrid Genetic Based Machine Learning and the Positive Definite Fuzzy Classifier domains of competence are extracted by means of the automatic extraction method.

From these intervals we obtain rules that describe both good or bad behaviors of the Fuzzy Rule Based Classification Systems mentioned, allowing us to characterize the response quality of the methods from the data set complexity metrics of a given data set. Thus, we can establish the domains of competence of the Fuzzy Rule Based Classification Systems considered, making it possible to establish when the method will perform well or poorly prior to its application.

J. Luengo is with the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, 18071, Granada, Spain. e-mail: julianlm@decsai.ugr.es.

F. Herrera is with the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, 18071, Granada, Spain. e-mail: herrera@decsai.ugr.es.

Index Terms

Classification, Data Complexity, Fuzzy Rule Based Systems, Domains of Competence

I. Introduction

Fuzzy Rule Based Classification Systems (FRBCSs) [23], [27] are widely employed due to their capacity to build a linguistic model interpretable to the users with the possibility of integrating different information such as that which comes from expert knowledge, from mathematical models or empiric measures.

New FRBCS models have been proposed on standard classification [21], [22], [31] and data streams [2] among others. They have been also applied widely including, but not limited to, the detection of intrusions [39], medical applications [1], [36], [40] and to the imbalanced data framework [16].

Issues such as the generality of the data, the inter-relationships among the variables and other factors are key for the prediction capabilities of the classifiers. An emergent field has arisen that uses a set of complexity measures [5] applied to quantify such particular aspects of the problem which are considered relevant to the classification task [19]. Studies of data complexity metrics applied to particular classification learning methods can be found in [6], [8], [17], [35].

The complexity of the data can be used to characterize FRBCSs' performance and it can be considered a new trend in the use of FRBCSs in pattern recognition. Different aspects of the data can be analyzed, relating those which indicate the easy and difficult problems for FRBCSs. No data complexity metrics have been analyzed together with FRBCSs up to now, except in our previous study [30].

In this work we propose a novel automatic extraction method for characterizing the domains of competence of FRBCSs by means of data complexity measures. These domains of competence characterize the range of data sets for which the FRBCSs obtains a good or bad performance on average. This allows the user to predict the FRBCS' behavior prior to its application, and to have in mind suitable parameter configurations or hints for improving the FRBCS. We consider twelve data complexity measures for each data set proposed by Ho and Basu [19] based on the overlaps in feature values from different classes; separability of

classes; and measures of geometry, topology, and density of manifolds. Each measure takes into account different properties of the data. Thus the proposal will be capable of analyzing different characteristics of the data and relating them to the performance of the FRBCSs.

In order to analyze and check the automatic extraction method, we consider two FRBCSs of different natures which have proven to perform well.

- The Positive Definite Fuzzy Classifier (PDFC) proposed by Chen and Wang [11] which is a Takagi-Sugeno based FRBCS.
- The Fuzzy Hybrid Genetic Based Machine Learning (FH-GBML) method proposed by Ishibuchi et al. [24] which is a Mamdani based FRBCS.

An initial set of 430 binary classification data sets created from real world problems is used to obtain the domains of competence of the two FRBCSs. Another set of 472 data sets is used to validate the domains of competence obtained by the automatic method.

Obtaining the FRBCSs' domains of competence by means of the automatic extraction method involves the following steps:

- It obtains intervals that describe when the FRBCSs perform well or poorly according to the data complexity values, using the initial 430 data sets.
- One rule for each interval is formulated, where some information and conclusions about the behavior of these methods can be stated.
- The individual rules are combined in order to improver their support and interpretability.
- Finally, two rules which discriminate the FRBCS's good or bad behavior each are obtained and validated with the extra 472 data sets.

The intervals which describe the performance of the FRBCSs are based on the following average values:

- Classification ratio, considering the average interval test accuracy, and its difference to the average global test accuracy (across all the initial 430 data sets) with respect to a specified threshold.
- Detection of the overfitting, by means of the difference between the training accuracy and test accuracy ratio.

From these intervals we define the domains of competence of the two FRBCSs. Both

descriptions are similar, which suggest that the FRBCSs are suitable for the same kind of problems, in spite of their different nature. Furthermore we compare the FRBCSs with three non-fuzzy classification methods: C4.5 [34] and Ripper [12] which are related to the FH-GBML rule induction; and SVM [13] which shares common learning steps with respect to PDFC. It can be seen that the domains of competence obtained generalize well and the FRBCSs are well characterized with respect to C4.5, Ripper and SVM.

The rest of this paper is organized as follows. In Section II the considered complexity measures are introduced as well as the most recent literature on the topic. Section III defines the domains of competence used, their related approach in the literature and their motivation. Section IV describes the automatic extraction method proposed in this work. In Section V we summarize the experimental framework, in which we show the data sets used, the FRBCSs' details and their parameters. In Section VI we include the experimental results obtained with the automatic extraction method and the domains of competence extracted, along with their analysis. Section VII contains the comparison between the FRBCSs and the crisp methods for the obtained domains of competence. Finally, in Section VIII some concluding remarks are made.

II. Data Complexity Measures

In the following subsections, we first present a short review of recent studies of Data Complexity Metrics (Subsection II-A), and then we describe the measures of overlapping (Subsection II-B), measures of separability of classes (Subsection II-C) and measures of geometry (Subsection II-D) used in our study.

A. Recent Studies on Data Complexity

As mentioned above, data complexity measures are a series of metrics that quantify data set characteristics which imply some difficulty for the classification task. In the following we gather several recent publications related to these complexity measures and their applications. They can show a picture of the most recent developments in the topic:

- In [19], Ho and Basu propose some complexity measures for binary classification problems, gathering metrics of three types: overlaps in feature values from different classes;

separability of classes; and measures of geometry, topology, and density of manifolds

- In [37], Singh offers a review of data complexity measures and proposes two new ones.
- In [8], Bernadó and Ho investigate the domain of competence of XCS by means of a methodology that characterizes the complexity of a classification problem by a set of geometrical descriptors.
- In [29], Li et al. analyze some omnivariate decision trees using the measure of complexity based on data density proposed by Ho and Basu.
- Baumgartner and Somorjai define specific measures for regularized linear classifiers in [6], using Ho and Basu's measures as reference.
- Sánchez et al. analyze the effect of data complexity on the nearest neighbors classifier in [35].
- Dong and Kothari propose in [14] a feature selection algorithm based on a complexity measure defined by Ho and Basu.
- Mollineda et al. in [32] extend some of Ho and Basu's measure definitions for problems with more than two classes. They analyze these generalized measures in two classic Prototype Selection algorithms and remark that Fisher's discriminant ratio is the most effective for Prototype Selection.
- Sang-Woon and Oommen [26] analyze how to use prototype selection in order to decrease the computation time of several data complexity measures, without severely affecting the outcome with respect to the complete data set.
- García et al. in [17] analyze the behavior of the evolutionary prototype selection strategy, considering a complexity measure for classification problems based on overlapping.
- Luengo and Herrera [30] characterized the behavior of the FH-GBML FRBCS by means of ad-hoc intervals of the data complexity measures from [19].

In our study we will use the twelve measures proposed in [19] which offer information for the FH-GBML and PDFC methods. They are summarized in Table I.

In the following Subsections we briefly describe these twelve measures, classified by their respective types.

TABLE I
DATA COMPLEXITY MEASURES

Type	Id.	Description
Measures of Overlaps in Feature Values from Different Classes	F1	maximum Fisher's discriminant ratio
	F2	volume of overlap region
	F3	maximum (individual) feature efficiency
Measures of Separability of Classes	L1	minimized sum of error distance by linear programming
	L2	error rate of linear classifier by Linear Programming
	N1	fraction of points on class boundary
	N2	ratio of average intra/inter class NN distance
	N3	error rate of 1NN classifier
Measures of Geometry, Topology and Density of Manifolds	L3	nonlinearity of linear classifier by linear programming
	N4	non-linearity of 1NN classifier
	T1	fraction of points with associated adherence subsets retained
	T2	average number of points per dimension

B. Measures of Overlaps in Feature Values from Different Classes

These measures are focused on the effectiveness of a single feature dimension in separating the classes, or the composite effects of a number of dimensions. They examine the range and spread of values in the data set within each class, and check for overlaps among different classes.

F1: *maximum Fisher's discriminant ratio*. Fisher's discriminant ratio for one feature dimension is defined as:

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where μ_1, μ_2, σ_1^2, σ_2^2 are the means and variances of the two classes respectively, in that feature dimension. We compute f for each feature and take the maximum as measure F1. For a multidimensional problem, not all features have to contribute to class discrimination. The problem is easy as long as there is only one discriminating feature. Therefore, we can just take the maximum f over all feature dimensions in discussing class separability.

F2: *volume of overlap region*. Let the maximum and minimum values of each feature f_i in class C_j be $\max(f_i, C_j)$ and $\min(f_i, C_j)$, then the overlap measure F2 is defined as

$$F2 = \prod_i \frac{MINMAX_i - MAXMIN_i}{MAXMAX_i - MINMIN_i}$$

where $i = 1, \ldots, d$ for a d-dimensional problem, and

$$MINMAX_i = MIN(max(f_i, C_1), max(f_i, C_2))$$

$$MAXMIN_i = MAX(min(f_i, C_1), min(f_i, C_2))$$

$$MAXMAX_i = MAX(max(f_i, C_1), max(f_i, C_2))$$

$$MINMIN_i = MIN(min(f_i, C_1), min(f_i, C_2))$$

F2 measures the amount of overlap of the bounding boxes of two classes. It is the product of a per-feature overlap ratio. The volume is zero as long as there is at least one dimension in which the value ranges of the two classes are disjointed.

F3: *maximum (individual) feature efficiency*. In a procedure that progressively removes unambiguous points falling outside the overlapping region in each chosen dimension [18], the efficiency of each feature is defined as *the fraction of all remaining points separable by that feature*. To represent the contribution of the most useful feature in this sense, we use the maximum feature efficiency as a measure. This measure considers only separating hyperplanes perpendicular to the feature axes. Therefore, even for a linearly separable problem, F3 may be less than 1 if the optimal separating hyperplane is oblique.

C. Measures of Separability of Classes

These measures provide indirect characterizations of class separability. They assume that a class is made up of single or multiple manifolds that form the support of the probability distribution of the given class. The shape, position and interconnectedness of these manifolds give hints on how well two classes are separated, but they do not describe separability by design. Some examples are shown as follows:

L1: *minimized sum of error distance by linear programming*. Linear classifiers can be obtained by a linear programming formulation proposed by Smith [38]. The method minimizes the sum of distances of error points to the separating hyperplane (subtracting a constant margin):

$$\text{minimize } \mathbf{a^t t}$$

$$\text{subject to } \mathbf{Z^t w + t \geq b}$$

$$\mathbf{t \geq 0}$$

where **a** and **b** are arbitrary constant vectors (both chosen to be 1), **w** is the weight vector to be determined, **t** is an error vector, and **Z** is a matrix where each column **z** is defined on an input vector **x** (augmented by adding one dimension with a constant value 1) and its class C (with value C_1 or C_2) as follows:

$$\begin{cases} \mathbf{z} = +\mathbf{x} & \text{if } C = C_1, \\ \mathbf{z} = -\mathbf{x} & \text{if } C = C_2. \end{cases}$$

The value of the objective function in this formulation is used as a measure. The measure has a zero value for a linearly separable problem. We should notice that this measure can be heavily affected by the presence of outliers in the data set.

L2: *error rate of linear classifier by Linear Programming* (LP). This measure is the error rate of the linear classifier defined for L1, measured with the training set. With a small training set this can be a severe underestimate of the true error rate.

N1: *fraction of points on class boundary*. This method constructs a class-blind minimum spanning tree over the entire data set, and counts the number of points incident to an edge going across the two classes. The fraction of such points over all points in the data set is used as a measure.

N2: *ratio of average intra/inter class Nearest Neighbor* (NN) *distance*. For each input instance x_p, we calculate the distance to its nearest neighbor within the class (intraDist(x_p)) and the distance to nearest neighbor of any other class (interDist(x_p)). Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example, i.e.,

$$N2 = \frac{\sum_{i=0}^{m} intraDist(x_i)}{\sum_{i=0}^{m} interDist(x_i)},$$

where m is the number of examples in the data set. This metric compares the within-class spread with the distances to the nearest neighbors of other classes. Low values of this metric suggest that the examples of the same class lie closely in the feature space. Large values indicate that the examples of the same class are disperse.

N3: *error rate of 1-NN classifier*. This is simply the error rate of a nearest-neighbor classifier measured with the training set. The error rate is estimated by the leave-one-out

method. The measure denotes how close the examples of different classes are. Low values of this metric indicate that there is a large gap in the class boundary.

D. Measures of Geometry, Topology and Density of Manifolds

These measures evaluate to what extent two classes are separable by examining the existence and shape of the class boundary. The contributions of individual feature dimensions are combined and summarized in a single score, usually a distance metric, rather than evaluated separately. Three measures from this family are described as follows:

L3: *nonlinearity of linear classifier by LP*. Hoekstra and Duin [20] proposed a measure for the nonlinearity of a classifier with respect to a given data set. Given a training set, the method first creates a test set by linear interpolation (with random coefficients) between randomly drawn pairs of points from the same class. Then the error rate of the classifier (trained by the given training set) on this test set is measured. Here we use a nonlinearity measure for the linear classifier defined for L1. This measure is sensitive to the smoothness of the classifier's decision boundary as well as to the overlap of the convex hulls of the classes.

N4: *nonlinearity of 1-NN classifier*. Following the same procedure presented for the L3 measure, in the case of N4 the error is calculated for a nearest neighbor classifier. This measure is for the alignment of the nearest-neighbor boundary with the shape of the gap or overlap between the convex hulls of the classes.

T1: *fraction of points with associated adherence subsets retained*. This measure originated from a work on describing shapes of class manifolds using the notion of adherence subsets in pretopology [28]. Simply speaking, it counts the number of balls needed to cover each class, where each ball is centered at a training point and grown to the maximum size before it touches another class. Redundant balls lying completely in the interior of other balls are removed. We normalize the count by the total number of points. In a problem where each point is closer to points of the other class than points of its own class, each point is covered by a distinctive ball of a small size, resulting in a high value of the measure.

T2: *average number of points per dimension*. This is a simple ratio of the number of points

in the data set over the number of feature dimensions, i.e.

$$T2 = \frac{m}{n},$$

where m is the number of examples in the data set, and n is the number of attributes of the data set.

III. Domains of Competence of Classifiers

In this section the motivation behind the proposal is presented in Subsection III-A. The concept of the domains of competence of the FRBCS used in this paper is given in Subsection III-B.

A. Approaches on the classifier performance characterization

To determine when a learning method will perform well or poorly is not a trivial task, considering accuracy as the performance measure. One primary indicative of the method's performance is the training accuracy. However, this is not always a precise measure. Figures 1 and 2 contain the accuracy results in training and test for FH-GBML, PDFC methods over all the initial 430 data sets (see Subsection V-A), plotted in ascending training accuracy value. We would like to point out how overfitting is continuously present in them. Therefore the necessity of other kinds of tools for characterizing the behavior of the methods appears.

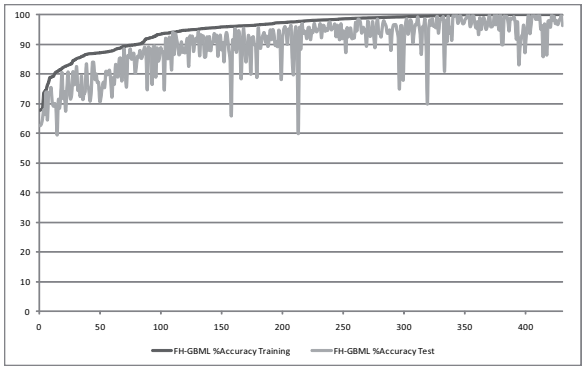

Fig. 1. Accuracy in Training/Test for FH-GBML sorted by training accuracy

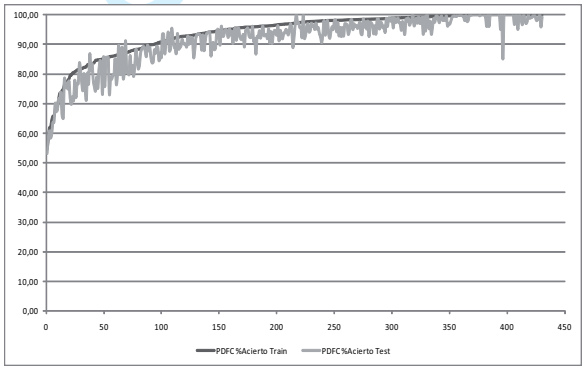

Fig. 2. Accuracy in Training/Test for PDFC sorted by training accuracy

One of the best-known approaches to predict the classifier performance is the Meta-Learning problem, which formalized this task [7], [9], [33]. Meta Learning is also intended to

select the best classifier for a given problem among several ones. A Meta Learning example most often involves a pair (Machine Learning problem instance, Machine Learning algorithm), labeled with the performance of the algorithm on the Machine Learning problem instance.

The two main problems of Meta Learning are the problem of the selection and the representation of Meta Learning examples.

- How to represent an Machine Learning problem instance was tackled using diverse descriptors, e.g. number of examples, number of attributes, percentage of missing values, landmarkers, etc. [33]. The difficulty is due to the fact that the descriptors must take into account the example distribution, which is not easily achieved in most cases.
- A second difficulty concerns the selection of the Machine Learning problem instances. Kalousis [25] indicates that the representativity of the problems and the perturbation induce strong biases in the Meta Learning classifier.

For these reasons among others, Meta Learning has achieved limited success.

We can also refer to the least known Phase Transition approach. The Phase Transition paradigm was initially developed to better understand the performances of Constraint Satisfaction algorithms indicating where the really hard problems are [10]. By means of this paradigm, a regular complexity landscape can be observed: the actual complexity is negligible in two wide regions, the YES and NO region, where the probability of satisfiability is respectively close to 1 and close to 0. These regions are separated by the so-called phase transition, where the hardest problems on average concentrate.

Using the Phase Transition, Baskiotis and Sebag [4] adapted the k-term DNF representation to classification problems, evaluating the C4.5 learning performance with respect to the underlying target concept. They defined C4.5 competence maps by means of generating boolean data sets of different characteristics (number of attributes, number of terms, etc.) based on a uniform distribution. C4.5 is then trained on these data sets and C4.5's error constitutes the complexity landscape (i.e. the competence map) using different data sets' configurations.

However these competence maps are only defined for binary attributes and they are based on the assumption of a uniformly distributed sample space, which is not usually true. Furthermore, the descriptive expressions obtained are not unique, hindering their interpretability.

The data complexity measures presented in the previous section are a recent and promising option and they can be used to carry out the characterization of the methods. One direct approach is to analyze the relationship between the data complexity value for a given data set and the performance obtained by the method. This approach does not suffer from the previous approaches' problems, as it is not dependent on a specific Machine Learning method either in the data distribution or the kind of data set attributes.

Following the criteria in [30] a series of rules are created to predict the regions *a priori* to the learning algorithm's behavior. Using enough data sets sorted by a particular data complexity measure, we can observe regions in which the method is performing noticeably well or poorly. In Figure 3, which shows the PDFC method's accuracy sorted by the N1 data complexity measure, good and bad results can be distinguished easily. However, this case does not usually apply when considering all the data complexity measures. Figure 4 shows an example of a data complexity measure in which no significant regions could be found for either good or bad behavior for the FH-GBML method.

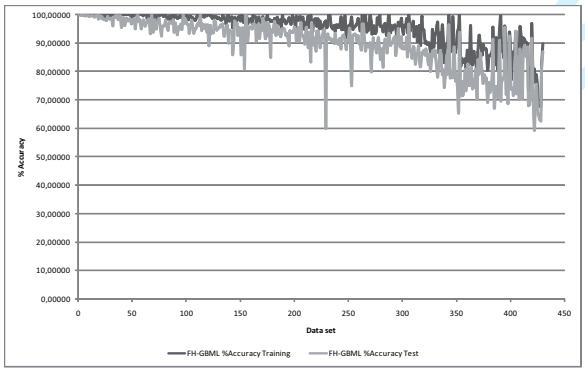

Fig. 3. Accuracy in Training/Test for PDFC sorted by N1

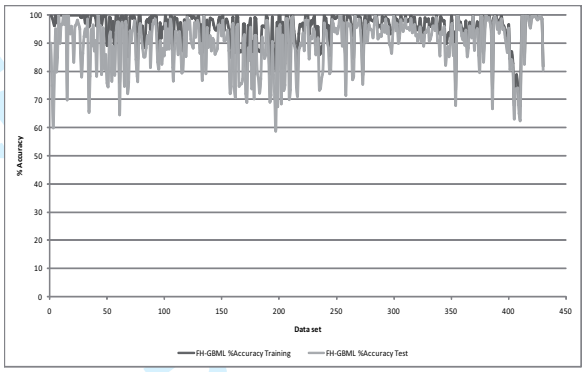

Fig. 4. Accuracy in Training/Test for FH-GBML sorted by T1

In this work an automatic extraction method capable of determining the domains of competence of the FRBCSs is proposed and presented in the next section. It is based on the good and bad behavior definitions initially explored in [30].

B. Domains of competence using data complexity measures

The concept of domains of competence involves the characterization of the range of problems for which a particular learning method is well suited or not. These problems share some aspects that explain the good or bad behavior of the method on them. There are some

previous approaches in the literature which define the domains of competence of a learning method based on the data complexity measures. As mentioned, this may be a key question when training a classification method takes a long time, due to the complexity of the learning algorithm or the size of the data. Problems denoted as difficult *a priori* may also require a different parameter configuration in respect to simpler ones. The data complexity measures also give hints on the nature of the data, and specific efforts can be made in order to improve the method's performance in the difficult regions.

Bernadó and Ho [8] initially defined the concept of domains of competence for the XCS classifier. Their domains of competence indicate the "region" of the complexity space of problems adequate to the learning method characteristics. Such a characterization could be useful for focusing the efforts of the learning algorithm's improvements on those difficult areas. In order to establish such limits, they used six of the twelve data complexity measures presented in Table I, and related the error rate of XCS to high or low values of the data complexity metrics. They also observed which kind of metrics best discriminate between difficult and easy problems for XCS.

In [30] their notion of domains of competence for the FH-GBML FRBCS is extended, using all of the twelve data complexity measures and eliminating the constraints on the low or high values. Specific intervals of the measures in which the FH-GBML method performs well or poorly (instead of relating its performance to "high" or "low" values) are extracted ad-hoc. These intervals were coded as rules which constitute the domains of competence of the learning method.

This concept of domains of competence can be related to other approaches in the literature presented in the previous subsection. The competence maps described by the Phase Transition paradigm can be considered as a limited version of these domains of competence. As mentioned in the previous subsection, they suffer from severe limitations that have not been overcome: the real-world data sets are rarely completely binary and uniformly distributed.

The domains of competence of two different classifiers cannot be compared in order to determine the best classifier. Only their shared strengths and weakness can be pointed out. The comparison between classifiers is directly related to the Meta Learning approach. As mentioned, many problems arise when trying to determine the best algorithm and no

satisfactory solution has been proposed. For these reasons the comparative model selection is out of the scope of this paper and we focus on the characterization of the domains of competence of individual FRBCSs. Only the independent recommendation of individual (or several) classifiers can be made on this basis.

It can be argued that, instead of extracting the domains of competence, it would be more useful to simply train and test the classifier over the considered data set. Nevertheless this same criticism can be applied to the Meta Learning and Phase Transition approaches and simply running the classifier is not always possible or desirable. For example, big data sets may cause the FRBCS to take too much time to train or the adequate parameter configuration may not be obvious. These issues can be addressed by means of the data complexity measures, as they work directly with the data and they give information on the data characteristics. This information cannot be extracted from the Meta Learning or Phase Transition approaches.

IV. Automatic Extraction Method

In the study performed in [30] an ad-hoc method used for extracting intervals for the FH-GBML was proposed. The intervals were extracted over the sorted data sets as described in Subsection III-A. This ad-hoc method was based on the selection of intervals of data complexity metrics' values, with some significance for the user according to a visual criteria for the good results.

Three main problems were found when dealing with the ad-hoc process of extracting intervals in [30]:

- The cut points which define the intervals were arbitrarily selected according to the graphics.
- It is possible to omit intervals with similar characteristics to the extracted ones. That is, the user is not using a formal description of the good or bad behavior intervals.
- The resultant characterization is subjective.

These issues can be tackled by the rigorous definition of the good and bad intervals, and by creating an automatic search method which extracts the domains of competence of the learning method. The automatic extraction method decides which data complexity measures are useful (if they contain significant intervals), and which measures are discarded (without providing

any interval for them). In Subsection IV-A the initial considerations for the definition of the good or bad behavior elements and intervals' is provided. Subsections IV-B and IV-C give the good and bad behavior elements and intervals definitions respectively. Finally the automatic extraction method is described in Subsection IV-D.

A. Initial considerations

The automatic extraction method analyzes a list of data sets where each complete data set has an associated performance measure of the FRBCS considered (typically the training and test accuracy rates) and the values of the twelve data complexity measures.

Definition 1. *Let $U = \{u_1, u_2, \ldots, u_n\}$ be a list of n different data sets. Each data set u_i has associated a tuple $T_{u_i} = (u_i^{tra}, u_i^{tst}, u_i^{F1}, u_i^{F2}, \ldots, u_i^{T2})$ where u_i^{tra} is the training accuracy value associated with the data set u_i for a specific FRBCS, u_i^{tst} is the test accuracy value associated with the data set u_i for the same FRBCS, and the set $CM_{u_i} = \{u_i^{F1}, u_i^{F2}, \ldots, u_i^{T2}\}$ contains the values for the twelve data complexity measures.*

Definition 2. *Given a list of data sets $U = \{u_1, u_2, \ldots, u_n\}$, we define the **average training accuracy** over U as $\bar{U}^{tra} = \frac{1}{n}\sum_{i=1}^{n} u_i^{tra}$ and the **average test accuracy** as $\bar{U}^{tst} = \frac{1}{n}\sum_{i=1}^{n} u_i^{tst}$.*

In order to define the domains of competence of an FRBCS, intervals of values of the data complexity measures need to be stated. The intervals are defined over the list U of data sets, sorting the list U by one data complexity measure CM_j of the twelve data complexity measures represented in the T tuple, defined as follows.

Definition 3. *A sorted list of data sets U_{CM_j} with respect to the data complexity CM_j is such that $\forall u_i^{CM_j}, u_j^{CM_j} \in U_{CM_j}; u_i^{CM_j} \leq u_j^{CM_j}$, if $i < j$.*

Definition 4. *Given a list of sorted data sets $U_{CM_j} = \{u_1, u_2, \ldots, u_n\}$ by the data complexity measure $CM_j \in T$, we consider an **interval** $V = \{u_i, u_{i+1}, \ldots, u_l\} \subseteq U_{CM_j}$ where the lower and upper bound values of V immediately follows:*

- $M_{low}(V) = \min_k\{u_k^{CM_j} \in V\} = u_i$.
- $M_{up}(V) = \max_k\{u_k^{CM_j} \in V\} = u_l$.

In our proposal, we distinguish between good and bad behavior elements (data sets), and good and bad behavior intervals. The latter ones are obtained using the former ones, and they are described in the next subsections.

B. Good and bad behavior elements

The distinction between good and bad behavior elements is based upon the absence or presence of overfitting, respectively, as well as the test accuracy value obtained. These are the two most common elements to determine when a classification algorithm performs well or poorly. The automatic method works with specific values, depending on the data sets, so we need to parameterize these two subjective indicators.

The former aspect is evaluated comparing the element's difference between training and test accuracy with respect to the global average. This global average difference is straightforwardly defined as

$$\bar{U}^{diff} = \frac{1}{n}\sum_{j=1}^{n} u_j^{tra} - u_j^{tst}. \tag{1}$$

An element with a difference between training and test accuracy above this average is considered to be overfitted, while the opposite case is not.

On the other hand any good behavior element u_i must present a minimum test accuracy value u_i^{test}, represented by the *minGoodElementTest*. By contrast a bad behavior element u_j shows a test accuracy value u_i^{test} under the same threshold *minGoodElementTest*.

With the aforementioned parameters, the definitions of the good and bad behavior elements are as follows.

Definition 5. *A **good behavior element** u_i is such that*

1) $u_i^{test} \geq minGoodElementTest$*; and*
2) $u_i^{tra} - u_i^{tst} \leq \frac{1}{n}\sum_{j=1}^{n} u_j^{tra} - u_j^{tst}$.

Definition 6. *A **bad behavior element** u_i is such that*

1) $u_i^{test} < minGoodElementTest$*; and*
2) $u_i^{tra} - u_i^{tst} > \frac{1}{n}\sum_{j=1}^{n} u_j^{tra} - u_j^{tst}$.

Due to the first item of both definitions, no element can be considered as a good behavior and bad behavior element simultaneously.

C. Good and bad behavior intervals

The good or bad behavior intervals V shows the good or bad performance of the classifier over a range of elements on average. Thus, for intervals, the definition of good or bad behavior is different with respect to individual elements, although they share the same underlying concepts: overfitting and test accuracy.

We consider the average difference across every element covered as a criteria to discriminate between good and bad behavior. The average interval V difference is defined as

$$\bar{V}^{diff} = \frac{1}{|V|} \sum_{u_j \in V}^{n} u_j^{tra} - u_j^{tst}. \tag{2}$$

A good interval must have a lower average difference than the global average $\bar{U}^{diff}$ defined in Equation (1). A bad interval must verify the opposite case.

We also establish a threshold between the average test accuracy of the interval $\bar{V}^{tst}$ and the global average $\bar{U}^{tst}$. An interval of good behavior must have an average test accuracy above this threshold plus $\bar{U}^{tst}$, while a bad behavior interval must verify the opposite case: $\bar{U}^{tst}$ minus the threshold. This condition reflects the behavior difference of the classifier with respect to the average.

In the case of good behavior intervals, we also establish that no element can have a test accuracy below $minGoodIntervalTest$ percent. The reason for this is to avoid very bad elements being covered by the interval due to the latter still having a good test average. This aspect is not crucial when defining bad intervals.

The definition of the good and bad behavior intervals using these parameters is as follows.

Definition 7. *An **interval of good behavior** $V = \{u_i, \ldots, u_j\}$ is such that*

- $\bar{V}^{diff} \leq \bar{U}^{diff}$*; and*
- $\bar{V}^{tst} \geq \bar{U}^{tst} + threshold$*; and*
- $\forall u_j \in U; u_j^{tst} \geq minGoodIntervalTest$

Definition 8. *An **interval of bad behavior** $V = \{u_i, \ldots, u_j\}$ is such that*

- $\bar{V}^{diff} > \bar{U}^{diff}$*; and*
- $\bar{U}^{tst} < \bar{U}^{tst} - threshold$*.*

D. Automatic Extraction Method Description

The automatic extraction method obtains a series of good or bad behavior intervals (as indicated in Definitions 7 and 8 respectively) from a list of data sets U. Each data set $u_i \in U$ has the associated tuple T containing the training and test accuracy values for a particular FRBCS (FH-GBML or PDFC in this case) and its twelve data complexity values.

In order to extract the good and bad behavior intervals, the automatic extraction method arranges the data sets in U by sorting them with one of the data complexity measures $CM_j, j = 1, \ldots, 12$, obtaining a new sorted list U_{CM_j}. Then the sorted list U_{CM_j} is explored from the lowest to highest values of CM_j. When a good or bad behavior element $u_i \in U_{CM_j}$ is found (Definitions 5 and 6 respectively), the exploration stops. The found element u_i is considered as an initial interval $V = \{u_i\}$, which is extended by adding the adjacent elements to u_i. This growing process continues while the interval V verifies the definitions of good or bad behavior intervals accordingly. A final interval $V = \{u_{i-r}, \ldots, u_i, \ldots, u_{i+s}\}; r, s \in \mathbb{N}$ is obtained when the corresponding definition is not met.

When all the possible intervals have been extracted, a final cleaning process is applied, in order to merge overlapped or slightly separated intervals of the same type, provided that the corresponding definition of a good or bad interval is maintained. Finally, a filtering is applied in order to avoid non-significant intervals. Those intervals with a support under 15% of the total data sets are discarded.

We present the main outline of the automatic extraction method described in Algorithm 1. The software can be downloaded from http://sci2s.ugr.es/DC-FRBCS-automatic-method.

The functions used in the algorithm are described as follows:

- $nextImportantGoodPoint(u_i, U)$: Looks for the index k of the next *good behavior point* u_k in the subset $V = \{u_i, \ldots, u_n\} \subseteq U$. If no good behavior point can be found it returns -1.

Algorithm 1 Automatic Extraction Method

Input: A list of data sets $U = \{u_1, u_2, \ldots, u_n\}$. Each data set u_i has associated a tuple T containing the training and test accuracy values for a particular learning method and its twelve data complexity values.

Output: A set of intervals G in which the learning method shows good or behavior, and a set of intervals B where the learning method shows bad behavior

Steps:

$G \leftarrow \{\}$
$B \leftarrow \{\}$
for each $CM_j \in T$ **do**
  //Sort the list U by each data complexity measure CM_j
  $U_{CM_j} \leftarrow sort(U, CM_j)$
  //Search for good behavior intervals
  $i \leftarrow 1$
  while $i < n$ **do**
    $pos \leftarrow$ *nextImportantGoodPoint*(u_i, U_{CM_j})
    if $pos \neq -1$ **then**
      $V \leftarrow$ *extendGoodInterval*(pos, U_{CM_j})
      $G \leftarrow G \cup \{V\}$
      $u_i \leftarrow M_{up}(V)$
    end if
  end while
  //Search for bad behavior intervals
  $i \leftarrow 1$
  while $i < n$ **do**
    $pos \leftarrow$ *nextImportantBadPoint*(u_i, U_{CM_j})
    if $pos \neq -1$ **then**
      $V \leftarrow$ *extendBadInterval*(pos, U_{CM_j})
      $B \leftarrow B \cup \{V\}$
      $u_i \leftarrow M_{up}(V)$
    end if
  end while
end for
//Merge and filter the intervals if necessary
$G \leftarrow mergeOverlappedIntervals(G)$
$G \leftarrow dropSmallIntervals(G)$
$B \leftarrow mergeOverlappedIntervals(B)$
$G \leftarrow dropSmallIntervals(B)$
return {G,B}

- $nextImportantBadPoint(u_i, U)$: Looks for the index k of the next *bad behavior point* u_k in the subset $V = \{u_i, \ldots, u_n\} \subseteq U$. If no bad behavior point can be found it returns -1.
- $extendGoodInterval(pos, U)$: From the reference point u_{pos} this method creates a new *interval of good behavior* $V = \{u_{pos-r}, \ldots, u_{pos}, \ldots, u_{pos+s}\} \subseteq U$, maintaining the element's order in U.
- $extendBadInterval(pos, U)$: From the reference point u_{pos} this method creates a new *interval of bad behavior* $V = \{u_{pos-r}, \ldots, u_{pos}, \ldots, u_{pos+s}\} \subseteq U$, maintaining the element's order in U.
- $mergeOverlappedIntervals(A)$: In this function, an interval V_k is dropped from A if $\exists V_m \in A; M_{up}(V_m) \geq M_{up}(V_k)$ and $M_{low}(V_m) \leq M_{low}(V_k)$. Moreover it tries to merge overlapped intervals $V_k, V_m; V_k \cap V_m \neq \emptyset$; or intervals separated by a maximum gap of

5 elements (data sets), provided that the new merged intervals satisfy Definition 7 and 8 of good or bad behavior respectively.

- $dropSmallIntervals(A)$: This function discards the intervals $V_k \in A$ which contains a number of data sets less than $0.15 \cdot n$.

The data sets used in this study are the same as were used in the ad-hoc study in [30]. Thus we have previous reference values for the threshold and the minimum acceptable accuracy ratio. We observe from [30] that the minimum threshold obtained by the ad-hoc rules was 6, and most of the data sets covered by the good intervals had 90% test accuracy. Consequently the values of the three parameters for the automatic method are as follows.

- $minGoodElementTest = 90$.
- $minGoodIntervalTest = 60$
- $threshold = 6$.

Please note that the parameters could be adjusted to the particular bunch of data sets used. That is, the concept of good or bad behavior varies according to the user and it can be tuned depending on the context.

V. Experimental Framework

In this section we first describe the data sets considered for the domains of competence extraction and evaluation in Subsection V-A. Next in Subsection V-B the basic notions of the the FH-GBML method and the PDFC method are depicted. Finally we show the parameters used for both FRBCSs and crisp classifiers used in Subsection V-C.

More detailed FRBCS descriptions can be obtained from the webpage http://sci2s.ugr.es/DC-FRBCS-automatic-method. All the data sets' generation details and their download packages are also available.

A. Data sets choice for the experimental study

In this paper a set of binary classification problems have been used. Initially, these problems were generated from pairwise combinations of the classes of 21 problems from the University of California, Irvine (UCI) repository [3]. We take each data set and extract the examples belonging to each class. Then we construct a new data set with the combination of the

examples from two different classes. This will result in a new data set with only 2 classes and the examples which have two such classes as output. In order to obtain additional data sets we also group the classes two by two.

If the data set proves to be linearly-separable, then we can classify it with a linear classifier with no error and such a data set would not be a representative problem. The complexity measure L1 indicates if a problem is linearly-separable when its value is zero, so every data set with an L1 value of zero is discarded. Finally, all these combinations resulted in 430 binary classification problems which are used as our training-bed for the automatic extraction method.

In order to validate the results obtained in our analysis, we have applied the same methodology of grouping the classes two by two to the *yeast, flare* and *car* data sets, due to their high number of classes. With these last three data sets we have obtained another 472 data sets used for validating the rules obtained in our analysis, for a total of 902 data sets.

B. Fuzzy Rule Based Classification Systems

Any classification problem consists of w training patterns $x_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, w$ from M classes where x_{pi} is the ith attribute value ($i = 1, 2, \ldots, n$) of the p-th training pattern.

In this work we use fuzzy rules in the following form:

$$\text{Rule } R_j: \text{ If } x_1 \text{ is } A_j^1 \text{ and } \ldots \text{ and } x_n \text{ is } A_j^n \text{ then Class = } C_j \text{ with } RW_j \tag{3}$$

where R_j is the label of the jth rule, $x = (x_1, \ldots, x_n)$ is an n-dimensional pattern vector, A_j^i is an antecedent fuzzy set, C_j is a class label or a numeric value, and RW_j is the rule weight. We always use triangular membership functions as antecedent fuzzy sets.

The FH-GBML method [24] consists of a Pittsburgh approach where each rule set is handled as an individual. It also contains a Genetic Cooperative-Competitive Learning approach (an individual represents a unique rule). The system defines 14 possible linguistic terms for each attribute which correspond to Ruspini's strong fuzzy partitions with two, three, four, and five uniformly distributed triangular-shaped membership functions. Furthermore, the system also uses "don't care" as an additional linguistic term.

The PDFC learning method [11] uses a Support Vector Machine (SVM) approach to build up the model. PDFC considers a fuzzy model with $m+1$ fuzzy rules of the form given in Equation (3) where A_j^k is a fuzzy set with membership function $a_j^k : \mathbb{R} \to [0,1]$, $RW_j = 1$ and $C_j = b_j \in \mathbb{R}$. Therefore PDFC is an FRBCS with constant THEN-parts. PDFC considers the use of membership functions generated from a reference function a^k through location transformation [15], like the symmetric triangle and the gaussian function. With these kind of membership functions a Mercer Kernel can be constructed and we can use the SVM algorithm to find an optimal separating hyperplane. Once we get such a hyperplane, the fuzzy rules can easily be extracted.

C. Parameters of the Methods

In this section we present the parameters used for all the data sets, adjusted empirically to obtain a good performance over all the 430 initial data sets. We use fixed parameters in all cases due to the necessity of analyzing the characterization of the methods independently of the procedure of adjusting the parameters, focusing only on the complexity of the data. The crisp classifiers used to compare with the FRBCSs are included as well, using the parameter configuration recommended by the authors. In Table II we have summarized the parameters used by the methods.

TABLE II
PARAMETERS USED BY THE METHODS (p IS THE NUMBER OF ATTRIBUTES IN THE DATA SET)

FRBCSs	Crips classifiers
FH-GBML	**Ripper**
Number of fuzzy rules: $5 \times p$ rules.	growing subset percentage = 0.66
Number of rule sets (N_{pop}): 200 rule sets.	K = 2
Crossover probability: 0.9.	**C4.5**
Mutation probability: $1/p$.	confidence level = 0.25
Number of replaced rules: All rules except the best-one (Pittsburgh-part, elitist approach), number of rules/5 (Michigan-part).	minimum item-sets per leaf = 2 pruning for the final tree = yes
Total number of generations: 1,000 generations.	**SVM**
Don't care probability: 0.5.	C =1
Probability of the application of the Michigan iteration: 0.5	tolerance Parameter = 0.001
PDFC	epsilon = 10^{-12}
C =100	Kernel type = Puk
d = 0.25	ω = 1.0
Positive definite function type: symmetric triangle	σ = 1.0

VI. EXPERIMENTAL STUDY

In this section the analysis of the automatic extraction method is presented. This study begins by presenting the intervals obtained by the automatic extraction method for the two

FRBCSs for the initial 430 data sets, and the the rules obtention from the intervals in Subsection VI-A. The analysis of the disjunctive rules and their conjunctive combination is presented in Subsection VI-B. Finally the rules which represent the domains of competence of the FRBCSs are validated with the extra amount of 472 data sets in Subsection VI-C.

In order to estimate the learning methods' accuracy we use a 10-fold cross validation scheme for the data sets presented in Subsection V-A. We take the average accuracy of training and test of the 10 partitions as a representative measure of the FRBCSs' performance. In Table III we summarize the global Training and Test accuracy obtained by the FRBCSs for the initial 430 data sets. All the results can be downloaded from the webpage http://sci2s.ugr.es/DC-FRBCS-automatic-method.

TABLE III
GLOBAL AVERAGE TRAINING AND TEST ACCURACY/STD. DEV. FOR FH-GBML AND PDFC OVER THE 430 DATA SETS

	Global % Accuracy Training Global Training std. dev.	Global % Accuracy Test Global Test std. dev.
FH-GBML	95.31% 6.01	89.88% 8.68
PDFC	94.08% 7.57	91.66% 8.65

A. *Intervals and Rule Extraction with the Automatic Extraction Method*

Once we have obtained the average accuracy in training and test of each FRBCS for every data set, we use the automatic extraction method described above in order to obtain a series of intervals of such data complexity measures in which each FRBCS has obtained significantly good or bad behavior with respect to the global performance. The precise concept of a good or bad behavior interval for the learning method has already been defined in Section IV.

In Table IV we summarize the intervals obtained by the automatic extraction method. We must point out that some of the intervals are shared among the FRBCSs, if not completely, at least in part of their supported data sets. It is also interesting to note how the automatic extraction method is capable of extracting different intervals for the different FRBCSs, due to the differences in performance and the common good and bad behavior definitions they share. The automatic extraction method has been capable of extracting more intervals of good behavior for the PDFC method than for FH-GBML. In the case of the intervals of bad behavior of the methods, the two FRBCS have the same number of bad intervals with similar data complexity measures.

TABLE IV
INTERVALS OBTAINED BY THE AUTOMATIC EXTRACTION METHOD FOR THE 430 DATA SETS

FH-GBML				PDFC			
Good behavior		Bad behavior		Good behavior		Bad behavior	
Measure	Range	Measure	Range	Measure	Range	Measure	Range
N1	[0.00117,0.1359]	F1	[0.2031,0.8689]	N1	[0.02759,0.08]	F3	[0.05674,0.3869]
N2	[0.00883,0.2976]	N1	[0.2582,0.66]	N2	[0.00883,0.1966]	N1	[0.2582,0.5714]
L1	[0.03021,0.1999]	N2	[0.5754,0.9256]	N3	[0.0,0.05714]	N2	[0.5754,0.9256]
L2	[0.0,0.09449]	N3	[0.07339,0.5426]	L1	[0.03021,0.1621]	N3	[0.07328,0.5426]
T1	[0.21,0.8779]	N4	[0.1227,0.4868]	L2	[0.0,0.07543]	N4	[0.1673,0.4868]
		L1	[0.5543,4.481]	T1	[0.21,0.8819]	L2	[0.3095,0.494]
		L2	[0.2822,0.494]			T1	[0.9539,0.9967]

We can derive a set of rules from the intervals. For these rules we consider the support of the rules over the 430 data sets, the average training and test accuracy of the covered data sets, and the difference with the global accuracy shown in Table III.

In Table V we depict the rules derived from the intervals obtained by the automatic extraction method for FH-GBML and Table VI shows the rules for PDFC.

TABLE V
RULES OBTAINED FOR FH-GBML FROM THE AUTOMATIC INTERVALS

Good behavior rules						
Id.	Range	% Support	% Training Accuracy	Training Difference	% Test Accuracy	Test Difference
R1+	N1 ∈ [0.00117,0.1359]	49.77	99.02	3.71	95.9	6.01
R2+	N2 ∈ [0.00883,0.2976]	37.44	98.97	3.66	95.94	6.06
R3+	N3 ∈ [0.0,0.05769]	43.72	99.25	3.94	95.88	6.00
R4+	L1 ∈ [0.03021,0.2201]	21.16	98.91	3.6	95.92	6.03
R5+	L2 ∈ [0.0,0.09449]	34.19	98.92	3.61	95.89	6
R6+	T1 ∈ [0.21,0.8779]	22.79	99.51	4.2	95.92	6.04
Bad behavior rules						
Id.	Range	% Support	% Training Accuracy	Training Difference	% Test Accuracy	Test Difference
R1-	F1 ∈ [0.2031,0.8689]	19.53	90.9	-4.41	83.85	-6.03
R2-	N1 ∈ [0.2582,0.66]	23.72	87.06	-8.25	78.1	-11.78
R3-	N2 ∈ [0.5754,0.9256]	16.51	87.47	-7.85	78.43	-11.46
R4-	N3 ∈ [0.07339,0.5426]	46.74	91.17	-4.14	83.87	-6.02
R5-	N4 ∈ [0.1227,0.4868]	37.91	90.26	-5.05	83.87	-6.01
R6-	L1 ∈ [0.5543,4.481]	31.40	91.22	-4.1	83.87	-6.01
R7-	L2 ∈ [0.2822,0.494]	22.09	89.68	-5.63	79.59	-10.3

Tables V and VI are organized with the following columns:

- The first column corresponds to the identifier of the rule for further references.
- The "Range" column presents the domain of the rule.
- The third column "Support" presents the percentage of data sets which verify the antecedent part of the rule.
- The column "% Training, Std. Dev." shows the average accuracy in training of all the examples which are covered by the rule. The standard deviation of the average training accuracy is also computed.
- The column "Training Difference" contains the difference between the training accuracy

TABLE VI
RULES OBTAINED FOR PDFC FROM THE AUTOMATIC INTERVALS

Good behavior rules						
Id.	Range	% Support	% Training Accuracy	Training Difference	% Test Accuracy	Test Difference
R1+	N1 ∈ [0.00117,0.1039]	39.07	98.81	4.73	97.66	6.01
R2+	N2 ∈ [0.00883,0.1966]	22.33	98.77	4.69	97.67	6.01
R3+	N3 ∈ [0.0,0.05714]	43.26	98.85	4.77	97.68	6.02
R4+	L1 ∈ [0.03021,0.1621]	15.12	98.64	4.56	97.7	6.04
R5+	L2 ∈ [0.0,0.07543]	28.37	98.58	4.5	97.66	6.01
R6+	T1 ∈ [0.21,0.8819]	23.26	98.8	4.72	97.7	6.04
Bad behavior rules						
Id.	Range	% Support	% Training Accuracy	Training Difference	% Test Accuracy	Test Difference
R1-	F3 ∈ [0.05674,0.3869]	23.02	89.23	-4.85	85.35	-6.3
R2-	N1 ∈ [0.2768,0.66]	22.56	83.86	-10.22	79.68	-11.98
R3-	N2 ∈ [0.5754,0.9256]	16.51	84.35	-9.73	80.4	-11.26
R4-	N3 ∈ [0.07328,0.5426]	46.98	89.07	-5.01	85.64	-6.02
R5-	N4 ∈ [0.1673,0.4868]	26.28	86.74	-7.34	83.4	-8.25
R6-	L2 ∈ [0.3095,0.494]	17.44	85.86	-8.22	80.9	-10.75
R7-	T1 ∈ [0.9539,0.9967]	34.19	89.18	-4.9	85.62	-6.04

of the rule and the training accuracy across all 430 data sets.

- The column "% Test, Std. Dev." shows the average accuracy in test of all the examples which are covered by the rule. The standard deviation of the average test accuracy is computed as well.
- The column "Test Difference" contains the difference between the test accuracy of the rule and the test accuracy across all 430 data sets.

As we can observe in these tables, the positive rules (denoted with a "+" symbol in their identifier) always show a positive difference with the global average, both in training and test accuracy. The negative ones (with a "-" symbol in their identifier) verify the opposite case.

The support of the rules shows us that we can characterize a wide range of data sets and obtain significant differences in accuracy. The differences for the good behavior rules in test are very close to 6, the *threshold* parameter value given in Section IV-D. This is not the case for all the bad behavior rules. This means that the overfitting limit is more determinant for the bad behavior rules, while in the good behavior ones it is less so.

An interesting fact is that the automatic method uses the same data complexity measures in order to characterize the good data sets for the two FRBCSs. Most of the data complexity measures used to describe the good behavior of the FRBCSs belong to the *measures of separability of classes* category. Therefore the two FRBCSs appear to share most of the complexity space in which they perform well.

The region of the complexity space in which the FRBCSs perform badly is very similar as well. However some differences appear, as FH-GBML needs the description given by the F1 and L1 measures while PDFC does not. On the other hand, PDFC bad behavior is described by F3 and T1 but PDFC is not.

With these simple and individual rules, an initial characterization of the data sets for which good or bad behavior is obtained for the FRBCSs can be considered. From these statements, we can conclude that the FRBCSs depend on very similar characteristics of the data when they perform well, but more particularities appear when they perform poorly. This is due to the close nature of the FRBCSs and it is well indicated by the automatic method.

B. Combination of the Individual Rules

The objective of this section is to analyze the effect of combining the rules. We consider the disjunctive combination (we use the *or* operator) of all the positive rules to obtain a single rule (Positive Rule Disjunction -PRD-). The same procedure is performed with all the negative ones so we obtain another rule (Negative Rule Disjunction -NRD-). The new disjunctive rules will be activated if any of the component rules' antecedents are verified. By means of merging the individual rules we can arrive at a more general description, with a wider support, of the behavior of the FRBCSs' methods.

The PRD and NRD rules may present overlapping in their support and a mutually exclusive description of the good and bad regions is desirable [30]. In order to tackle this issue we consider the conjunctive operator *and* and the difference operator *and not* between the PRD and NRD rules. The difference will remove the data sets for which the FRBCSs present good or bad behavior from the disjunctive negative or positive rules, respectively. That is, by means of the difference we try to remove the data sets of the opposite type from the considered disjunctive rule. Thus we obtain three different kinds of intersection and an extra region:

- Intersection of positive disjunction *and* the negative disjunction (PRD$\wedge$NRD).
- Intersection of positive disjunction *and not* the negative disjunction (PRD$\wedge\neg$NRD).
- Intersection of negative disjunction *and not* the positive disjunction (NRD$\wedge\neg$PRD).
- *Not characterized* region, in which no rule covers its data sets.

TABLE VII
DISJUNCTIVE RULES FROM ALL SIMPLE RULES FOR FH-GBML AND PDFC

FH-GBML						
Id.	Range	% Support	% Training	Training	% Test	Test
PRD	If R1+ or R2+ or R3+ or R4+ or R5+ or R6+ then good behavior	58.60	98.64	3.33	94.76	4.88
NRD	If R1- or R2- or R3- or R4- or R5- or R6- or R7- then bad behavior	64.65	93.27	-2.04	87.02	-2.86
PRD ∧NRD	If PRD and NRD then good behavior	24.41	97.98	2.67	93.98	4.10
PRD ∧¬ NRD	If PRD and not NRD then good behavior	34.19	99.12	3.81	95.32	5.44
NRD∧¬PRD	If NRD and not PRD then bad behavior	36.81	90.42	-4.89	82.79	-7.09
not characterized	If not (PRD or NRD) then bad behavior	1.06	96.61	1.30	89.33	-0.55
PDFC						
Id.	Range	% Support	% Training Accuracy	Training Difference	% Test Accuracy	Test Difference
PRD	If R1+ or R2+ or R3+ or R4+ or R5+ or R6+ then good behavior	53.49	98.28	4.20	96.91	5.25
NRD	If R1- or R2- or R3- or R4- or R5- or R6- or R7- then bad behavior	61.40	91.09	-2.99	88.06	-3.60
PRD ∧NRD	If PRD and NRD then good behavior	16.28	96.99	2.91	95.63	3.97
PRD ∧¬ NRD	If PRD and not NRD then good behavior	37.21	98.85	4.77	97.47	5.81
NRD∧¬PRD	If NRD and not PRD then bad behavior	45.12	88.96	-5.12	85.33	-6.33
not characterized	If not (PRD or NRD then good behavior	1.40	98.51	4.43	94.79	3.13

In Table VII we depict the new collective rules for the FRBCSs. From them, we can point out the following for the two FRBCSs:

- The Positive Rule Disjunction (PRD) offers a high support for both the two FRBCSs, and it also gives a good training and test accuracy (over 97% and 91% respectively).
- The Negative Rule Disjunction (NRD) obtains a wide support as well (over 62%). However, the differences in both training and test have decreased due to this increment in support with respect to the single rules of bad behavior.
- The Positive and Negative Rule Disjunction (PRD∧NRD) is more specific than PRD in isolation. It is also similar to PRD in the training and test accuracy difference. This rule obtains positive differences in training and test accuracy, representing the good data sets for the FRBCSs covered by the rules of bad behavior.
- The Positive and Not Negative Rule Disjunction (PRD∧¬NRD) has a lower support than PRD∧NRD except in the case of the PDFC method. Its difference is higher than PRD and PRD∧NRD rules, since the data sets with low accuracy for the FRBCSs present in

PRD have been removed by PRD∧NRD.

- The Negative and Not Positive Rule Disjunction (NRD∧¬PRD) is a good rule to describe the bad behavior of the FRBCSs. It has a high support and a high difference in both training and test sets. When removing the good data sets of PRD∧NRD, the NRD∧¬PRD rule becomes more accurate.
- The data sets not characterized by either the PRD rule or the NRD rule always present a small difference from the global accuracy both in training and test. Therefore they cannot be classified as good or bad, but they constitute a low percentage of the data sets.

From all the disjunctive and new conjunctive rules, we can present PRD as a representative description of good data sets, and NRD∧¬PRD as a representative description for bad data sets, when using the FRBCSs. As these rules are mutually exclusive, we can consider three blocks of data sets with their respective support. In Figures 5 and 6 we depict the three block regions for the FH-GBML method and the PDFC method respectively. On the left, the data sets covered by PRD are depicted. On the right the data sets covered by NRD∧¬PRD are plotted. The small regions in the center correspond to the not characterized data sets.

From Figures 5 and 6 we can observe that more than 99% of the analyzed data sets are characterized in both cases. Using the ad-hoc extraction method in [30] the PRD and NRD∧¬PRD rules covered 75% of the 430 data sets for the FH-GBML method, approximately 25% less. Thus, the use of the automatic extraction method clearly outperforms the *ad-hoc* approach, improving the characterization of the data sets using the data complexity measures. Hence the domains of competence of *good behavior* and *bad behavior* characterization for the FRBCSs can be achieved by the PRD and NRD∧¬PRD rules.

C. Validation of the Collective Rules

Once we have obtained a set of descriptive rules of the domains of competence for the two FRBCSs with the automatic extraction method, their domains of competence have been established. In order to validate and evaluate how well these domains of competence generalize we should use an extra bunch of data sets which have not been used previously.

As described in Subsection V-A, we validate the PRD and NRD∧¬PRD rules using a set

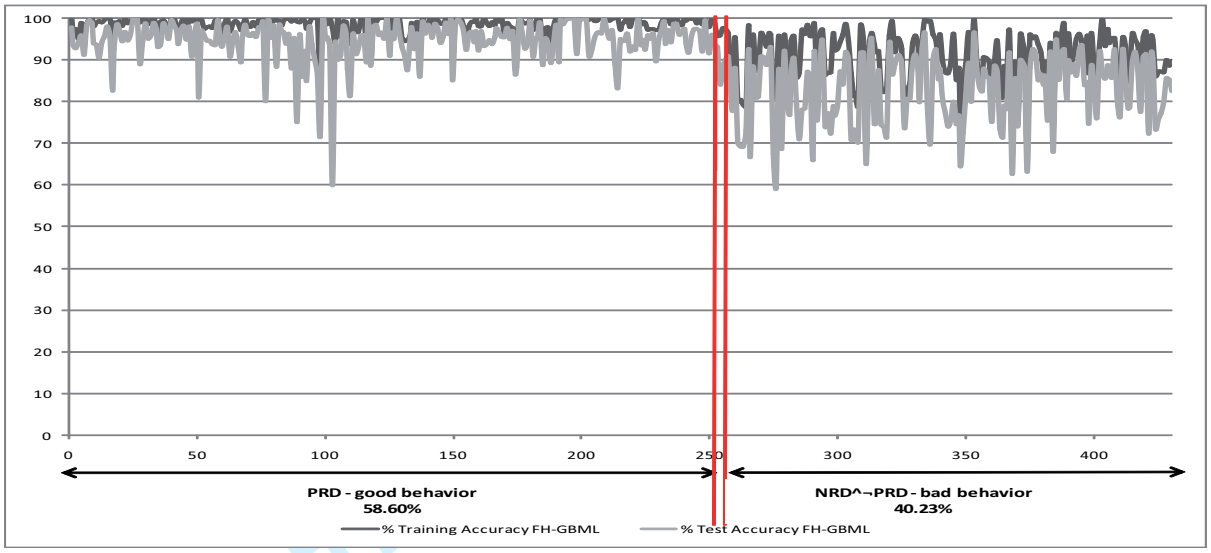

Fig. 5. FH-GBML block representation for PRD, not characterized and NRD∧¬PRD covered data sets

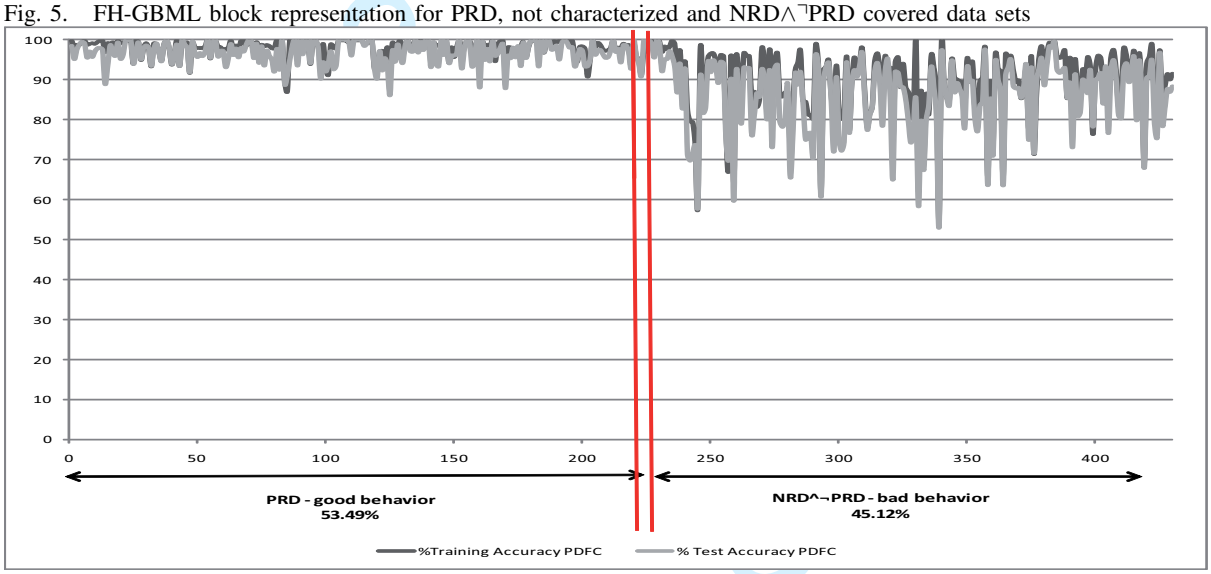

Fig. 6. PDFC block representation for PRD, not characterized and NRD∧¬PRD covered data sets

of 472 fresh data sets. In Table VIII we summarize the average training and test accuracy values for each classifier.

TABLE VIII
GLOBAL AVERAGE TRAINING AND TEST ACCURACY/STD. DEV. FOR FH-GBML AND PDFC OVER THE VALIDATION DATA SETS

	Global % Accuracy Training Global Training std. dev.	Global % Accuracy Test Global Test std. dev.
FH-GBML	91.33% 7.61	89.03% 8.44
PDFC	91.87% 7.49	90.19% 8.19

We present Table IX with the average training and test accuracy of the data set covered by the PRD and NRD∧¬PRD rules for each FRBCS, and the difference from the global average of the new 472 data sets.

TABLE IX
VALIDATION RESULTS FOR PRD AND NRD∧¬PRD RULES

FH-GBML					
Id.	% Support	% Training Accuracy	Training Difference	% Test Accuracy	Test Difference
PRD	52.54	95.62	4.29	94.03	5.01
NRD∧¬PRD	47.46	86.57	-4.76	83.50	-5.53
not characterized	0	-	-	-	-
PDFC					
Id.	% Support	% Training Accuracy	Training Difference	% Test Accuracy	Test Difference
PRD	47.46	95.79	3.92	94.18	3.99
NRD∧¬PRD	52.54	87.31	-4.56	84.38	-5.81
not characterized	0	-	-	-	-

The two blocks figure representation of the data of Table IX for the FH-GBML method and the PDFC method are depicted in Figures 7 and 8 respectively.

From these results we can observe the following:

- The good behavior characterized by PRD is maintained in this new set of data sets. The bad behavior characterized by NRD∧¬PRD is also obtained.
- All the validation data sets are characterized, showing the good generalization of the domains of competence obtained.

For the FRBCSs considered, their particular PRD and NRD∧¬PRD rules has a support of around 50% for each one, which indicates a good balance in the characterization of the good and bad domains of competence obtained. The definitions of good and bad behavior intervals given in Subsection IV can be applied to different methods to obtain robust rules. The PRD and NRD∧¬PRD rules obtained from the use of the automatic extraction method can predict the behavior of the FRBCSs, defining their domains of competence.

VII. COMPARISON OF THE DOMAINS OF COMPETENCE WITH CRISP MODELS

In this section we analyze the domains of competence of the FRBCSs in comparison with three crisp models related to them. We consider the C4.5 [34] and Ripper [12] as a crisp methods related to FH-GBML as both of them are rule learners. An SVM algorithm [13] is compared with the PDFC algorithm, as the latter adjusts its rules using the same procedure. The parameters used for their execution are shown in Subsection V-C.

Table X summarizes the average test accuracy of FH-GBML versus C4.5 and Ripper; and PDFC versus SVM for the 472 validation data sets. It depicts the global average test

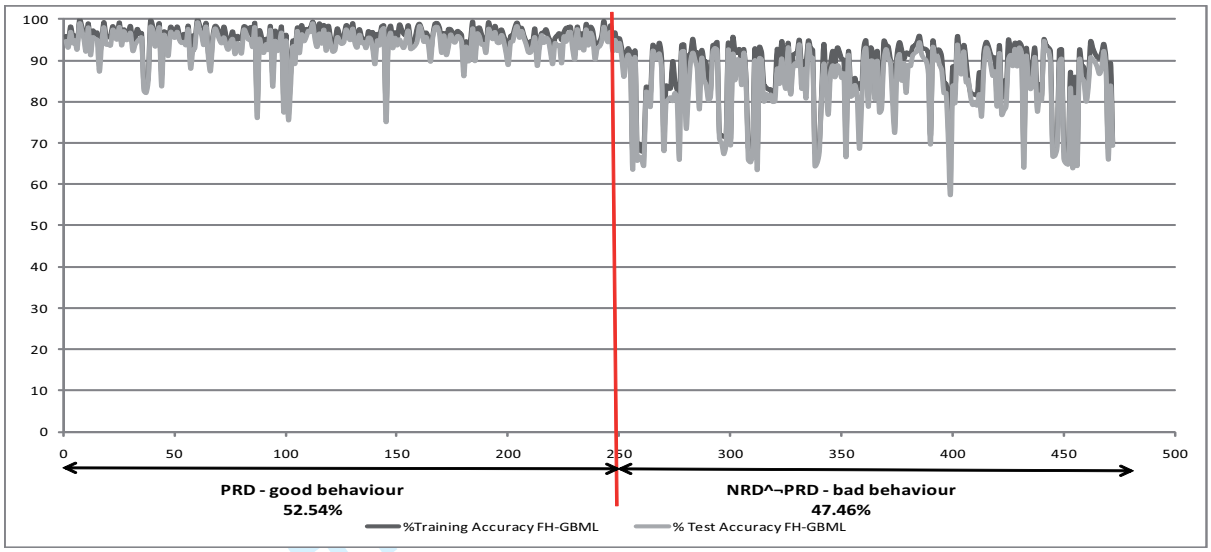

Fig. 7. FH-GBML block representation for PRD and NRD∧¬PRD for validation data sets

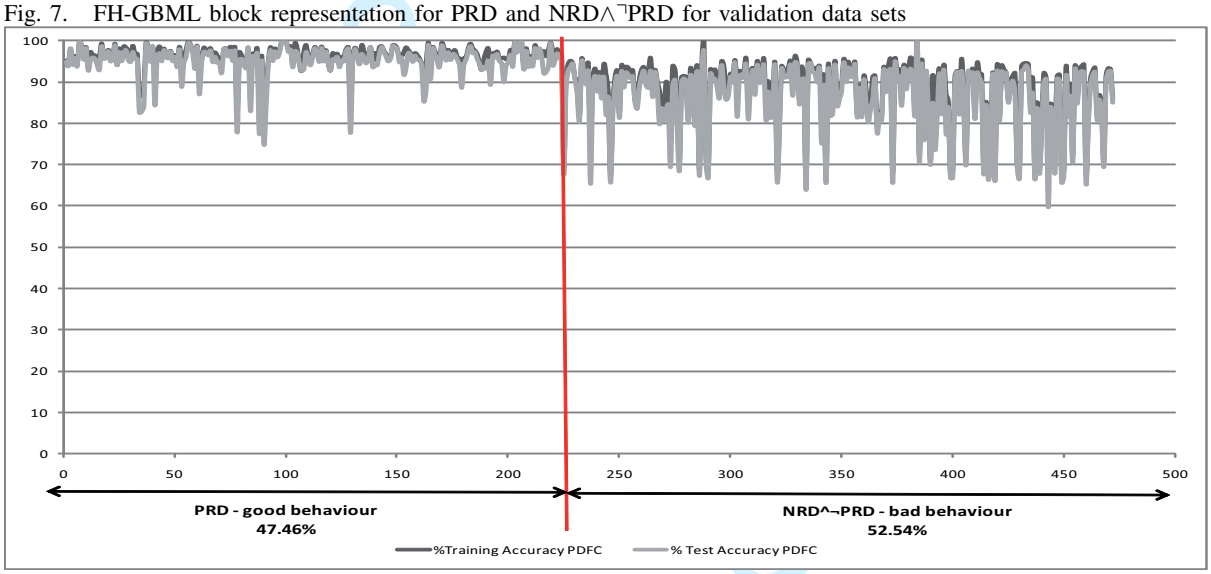

Fig. 8. PDFC block representation for PRD and NRD∧¬PRD for validation data sets

accuracy of each method, as well as the average test accuracy in each good and bad domain of competence of the FRBCSs.

C4.5 has a better average performance than FH-GBML while Ripper has a lower average test accuracy. However the differences between the methods are small, particularly for C4.5 and FH-GBML. Both C4.5 and Ripper have homogeneous behavior, presenting improvement in the FH-GBML domain of competence for good data sets and behaving worse in the opposite case. Thus the three rule based methods (both crisp and FRBCSs) show common behavior and a comparison between them cannot be established, as indicated in Subsection III-B.

TABLE X
AVERAGE TEST ACCURACY FOR FH-GBML VS. C4.5 AND RIPPER; AND PDFC VS. SVM IN THEIR DOMAINS OF COMPETENCE

	Test accuracy		
	FH-GBML	C4.5	Ripper
Global	89.03	89.18	86.17
PRD	94.03	94.11	91.61
NRD∧¬PRD	83.50	83.73	80.15
	Test accuracy		
	PDFC	SVM	
Global	90.19	90.16	
PRD	95.06	94.75	
NRD∧¬PRD	85.78	86.02	

PDFC and SVM methods perform approximately the same on average. In the case of considering the data sets covered by the PRD rule, this difference is increased in favor of PDFC. For the data sets covered by the NRD∧¬PRD rule, SVM performs better than PDFC. Thus for the PDFC method a different behavior is shown with respect to SVM.

The rule based methods, C4.5 and Ripper, show a behavior concordant with the rule obtained for the FRBCSs: better performance for the PRD rule, worse performance for the NRD∧¬PRD rule. The approximate model, SVM, shows an opposite behavior when compared with these rules. Thus rule based methods are characterized by similar domains of competence. This is not verified for the approximate method (SVM) which has different nature.

VIII. CONCLUDING REMARKS

We have proposed a new automatic extraction method to characterize the domains of competence of FRBCSs. In order to analyze it, we have performed a study over a large set of binary data sets with two FRBCSs, formally named the FH-GBML method and the PDFC method. First we have computed twelve data complexity measures for each data set. Next we have used the automatic extraction method and we have obtained a different set of intervals for each FRBCS in which the method's performance is significantly good or bad. Then we have built descriptive rules from these intervals and we have studied the interaction between the proper rules.

We have obtained two rules which are precise to describe the both good and bad performance of each of the two FRBCSs. We have validated these rules with an extra benchmark of independent data sets in order to check their generalization and prediction capabilities. The

definition of the domains of competence of good or bad behavior is therefore provided and it has been compared with three crisp methods, showing that these domains of competence are not only related between FRBCSs, but also with crisp rule based models due to their related nature. Finally, we present the possibility of determining automatically which data sets would prove to be good or bad for the two FRBCSs prior to their execution, using the data complexity measures.

We must point out that this is a study that uses two specific methods. These two methods are very different in their nature so we can confirm the capabilities of this methodology. This work presents a new challenge that could be extended to other learning methods, to automatically analyze their domains of competence, and to develop new measures which could provide us with more information on the behavior of FRBCSs for pattern recognition.

ACKNOWLEDGMENT

Supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01. J. Luengo holds a FPU scholarship from Spanish Ministry of Education and Science.

REFERENCES

[1] M. R. Akbarzadeh-Totonchi and M. Moshtagh-Khorasani, "A hierarchical fuzzy rule-based approach to aphasia diagnosis," *Journal of Biomedical Informatics*, vol. 40, no. 5, pp. 465–475, 2007.

[2] P. P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1462–1475, 2008.

[3] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://archive.ics.uci.edu/ml/

[4] N. Baskiotis and M. Sebag, "C4.5 competence map: a phase transition-inspired approach," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM, 2004, p. 8.

[5] M. Basu and T. K. Ho, *Data Complexity in Pattern Recognition (Advanced Information and Knowledge Processing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[6] R. Baumgartner and R. L. Somorjai, "Data complexity assessment in undersampled classification of high-dimensional biomedical data," *Pattern Recognition Letters*, vol. 12, pp. 1383–1389, 2006.

[7] H. Bensusan and A. Kalousis, "Estimating the predictive accuracy of a classifier," in *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*. London, UK: Springer-Verlag, 2001, pp. 25–36.

[8] E. Bernadó-Mansilla and T. K. Ho, "Domain of competence of XCS classifier system in complexity measurement space," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 82–104, 2005.

[9] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*, ser. Cognitive Technologies. Springer, January 2009.

[10] P. Cheeseman, B. Kanefsky, and W. M. Taylor, "Where the really hard problems are," in *IJCAI'91: Proceedings of the 12th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, pp. 331–337.

[11] Y. Chen and J. Z. Wang, "Support vector learning for fuzzy rule-based classification systems," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 6, pp. 716–728, 2003.

[12] W. Cohen, "Fast effective rule induction," in *Machine Learning: Proceedings of the Twelfth International Conference*, 1995, pp. 1–10.

[13] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[14] M. Dong and R. Kothari, "Feature subset selection using a new definition of classificabilty," *Pattern Recognition Letters*, vol. 24, pp. 1215–1225, 2003.

[15] D. Dubois and H. Prade, "Operations on fuzzy numbers," *International Journal of Systems Sciences*, vol. 9, pp. 613–626, 1978.

[16] A. Fernández, S. García, M. J. del Jesús, and F. Herrera, "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets," *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2378–2398, 2008.

[17] S. García, J. R. Cano, E. Bernadó-Mansilla, and F. Herrera, "Diagnose of effective evolutionary prototype selection using an overlapping measure," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 8, pp. 2378–2398, 2009.

[18] T. K. Ho and H. S. Baird, "Pattern classification with compact distribution maps," *Computer Vision and Image Understanding*, vol. 70, no. 1, pp. 101–110, 1998.

[19] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE Transactions on Pattern Analysins and Machine Intelligence*, vol. 24, no. 3, pp. 289–300, 2002.

[20] A. Hoekstra and R. P. Duin, "On the nonlinearity of pattern classifiers," in *ICPR '96: Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume IV-Volume 7472*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 271–275.

[21] J. Hühn and E. Hüllermeier, "FR3: A fuzzy rule learner for inducing reliable classifiers," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 1, pp. 138–149, 2009.

[22] ——, "FURIA: an algorithm for unordered fuzzy rule induction," *Data Mining and Knowledge Discovery*, vol. 19, no. 3, pp. 293–319, 2009.

[23] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*. Springer-Verlag New York, Inc., 2004.

[24] H. Ishibuchi, T. Yamamoto, and T. Nakashima, "Hybridization of fuzzy GBML approaches for pattern classification problems," *IEEE Transactions on System, Man and Cybernetics B*, vol. 35, no. 2, pp. 359–365, 2005.

[25] A. Kalousis, "Algorithm selection via meta-learning," Ph.D. dissertation, Université de Geneve, 2002.

[26] S.-W. Kim and B. J. Oommen, "On using prototype reduction schemes to enhance the computation of volume-based inter-class overlap measures," *Pattern Recognition*, vol. 42, no. 11, pp. 2695–2704, 2009.

[27] L. Kuncheva, *Fuzzy Classifier Design*. Springer, Berlin, 2000.

[28] F. Lebourgeois and H. Emptoz, "Pretopological approach for supervised learning," in *ICPR '96: Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume IV-Volume 7472*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 256–260.

[29] Y. Li, S. Member, M. Dong, R. Kothari, and S. Member, "Classifiability-based omnivariate decision trees," *IEEE Transactions on Neural Networks*, vol. 16, pp. 1547–1560, 2005.

[30] J. Luengo and F. Herrera, "Domains of competence of fuzzy rule based classification systems with data complexity measures: A case of study using a fuzzy hybrid genetic based machine learning method," *Fuzzy Sets and Systems*, vol. 161, no. 1, pp. 3–19, 2010.

[31] E. G. Mansoori, M. J. Zolghadri, and S. D. Katebi, "SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 4, pp. 1061–1071, 2008.

[32] R. A. Mollineda, J. S. Sánchez, and J. M. Sotoca, "Data characterization for effective prototype selection," in *Proc. of the 2nd Iberian Conf. on Pattern Recognition and Image Analysis*. Springer, 2005, pp. 27–34.

[33] B. Pfahringer, H. Bensusan, and C. G. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms," in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 743–750.

[34] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo–California: Morgan Kaufmann Publishers, 1993.

[35] J. S. Sánchez, R. A. Mollineda, and J. M. Sotoca, "An analysis of how training data complexity affects the nearest neighbor classifiers," *Pattern Analysis & Applications*, vol. 10, no. 3, pp. 189–201, 2007.

[36] G. Schaefer, M. Zavisek, and T. Nakashima, "Thermography based breast cancer analysis using statistical features and fuzzy classification," *Pattern Recognition*, vol. 42, no. 6, pp. 1133–1137, 2009.

[37] S. Singh, "Multiresolution estimates of classification complexity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1534–1539, 2003.

[38] F. W. Smith, "Pattern classifier design by linear programming," *IEEE Transactions on Computers*, vol. 17, no. 4, pp. 367–372, 1968.

[39] C. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," *Pattern Recognition*, vol. 40, no. 9, pp. 2373–2391, 2007.

[40] S. A. Vinterbo, E.-Y. Kim, and L. Ohno-Machado, "Small, fuzzy and interpretable gene expression based classifiers," *Bioinformatics*, vol. 21, no. 9, pp. 1964–1970, 2005.

3. Analysis of Over-sampling and Under-sampling approaches for Imbalanced Problems using Data Complexity Measures

The journal papers associated to this part are:

- J. Luengo, A. Fernandez, S. García, F. Herrera, Addressing Data Complexity for Imbalanced Data Sets: Analysis of SMOTE-based Oversampling and Evolutionary Undersampling. Soft Computing, doi:10.1007/s00500-010-0625-8, in press (2011).
  - Status: **In press**.
  - Impact Factor (JCR 2009): 1.328.
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 51 / 103.
  - Subject Category: Computer Science, Interdisciplinary Applications. Ranking 41 / 95.

Soft Comput
DOI 10.1007/s00500-010-0625-8

FOCUS

Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling

Julián Luengo · Alberto Fernández · Salvador García · Francisco Herrera

Abstract In the classification framework there are problems in which the number of examples per class is not equitably distributed, formerly known as imbalanced data sets. This situation is a handicap when trying to identify the minority classes, as the learning algorithms are not usually adapted to such characteristics. An usual approach to deal with the problem of imbalanced data sets is the use of a preprocessing step. In this paper we analyze the usefulness of the data complexity measures in order to evaluate the behavior of undersampling and oversampling methods. Two classical learning methods, C4.5 and PART, are considered over a wide range of imbalanced data sets built from real data. Specifically, oversampling techniques and an evolutionary undersampling one have been selected for the study. We extract behavior patterns from the results in the data complexity space defined by the measures, coding them as intervals. Then, we derive rules from the intervals that describe both good or bad behaviors of C4.5 and PART for the different preprocessing approaches, thus obtaining a complete characterization of the data sets and the differences between the oversampling and undersampling results.

J. Luengo (✉) · F. Herrera
Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain
e-mail: julianlm@decsai.ugr.es

F. Herrera
e-mail: herrera@decsai.ugr.es

A. Fernández · S. García
Department of Computer Science, University of Jaén, 23071 Jaén, Spain
e-mail: alberto.fernandez@ujaen.es

S. García
e-mail: sglopez@ujaen.es

Published online: 20 June 2010

Keywords Classification · Evolutionary algorithms · Data complexity · Imbalanced data sets · Oversampling · Undersampling · C4.5 · PART

1 Introduction

The problem of imbalanced classes is one of the problems that emerged when Machine Learning (ML) reached maturity, being a widely used technology in the world of business, industry, and scientific research. Its importance grew as researchers realized that the analyzed data sets contained many more instances or examples from a class or classes with respect to the remaining ones (Chawla et al. 2004), and the obtained classification models performed below the desired threshold in the minority classes. Currently it is considered as a challenge by the Data Mining Community (Yang and Wu 2006).

The main handicap of this type of problem is that standard learning algorithms minimize a global measure of error, and this supposes a bias towards the majority class (Sun et al. 2009). Hence, to tackle this issue, the use of preprocessing techniques is a good solution in order to balance the training set before the learning process (Batista et al. 2004; Estabrooks et al. 2004; He and Garcia 2009).

On the other hand, it is well known that the prediction capabilities of the classifiers are also dependent on the problem's characteristics as well. An emergent field, that uses a set of complexity measures applied to the problem to describe its difficulty, has recently arisen. These measures try to capture different aspects or sources of complexity which are considered complicated to the classification task (Basu and Ho 2006). Studies of data complexity metrics applied to particular classification's algorithms can be found (Basu and Ho 2006; Bernadó-Mansilla and Ho 2005;

Baumgartner and Somorjai 2006; Sánchez et al. 2007; García et al. 2009c).

Our objective is to show that the data complexity measures are adequate to analyze the effect of the preprocessing in imbalanced data for classification. We will consider two main preprocessing approaches: oversampling and undersampling of the data. We will identify the regions in the data complexity space in which the preprocessing works well, and the bad performance regions as well. In a related approach (García et al. 2008) the relationship between the Imbalance Ratio (IR) (Orriols-Puig and Bernadó-Mansilla 2008) and the overlapping of the class labels with respect to the performance of several learning methods was studied. However, no preprocessing approach was analyzed in this study.

In order to analyze the oversampling and undersampling by means of the data complexity measures, we will use the "Synthetic Minority Over-sampling Technique" (SMOTE) and its variant with the Wilson's Edited Nearest Neighbor Rule (ENN) as representative oversampling preprocesing methods. SMOTE is a classical oversampling method, whereas SMOTE-ENN was shown in Batista et al. (2004) to achieve a very good behavior with the C4.5 decision tree. The Evolutionary Undersampling with CHC (EUSCHC) method proposed by García and Herrera (2009a) will be included as a representative evolutionary undersampling approach. It has been proved to be very competitive with respect to SMOTE and SMOTE-ENN, and to be the best among other representative techniques from the family of undersampling as shown in their study.

The effect of these three preprocessing techniques will be analyzed with respect to two well-known learning methods. The first one is the C4.5 decision tree (Quinlan 1993), which has been used in many recent analyses of imbalanced data (Su and Hsiao 2007; García et al. 2009b; Drown et al. 2009). The second one is the PART algorithm (Frank and Witten 1998) also used by García et al. (2009b) in the imbalanced data framework.

Following the methodology proposed by Luengo and Herrera (2010), three of the data complexity measures proposed by Ho and Basu (2002) are informative in order to create intervals of their values over the data sets in which C4.5 and PART perform well or bad on average after applying each preprocessing technique. We will use a large collection of data sets with different degrees of imbalance from the UCI repository (Asuncion and Newman 2007) in order to sample the data complexity space. Then we will formulate rules for such intervals, comparing the support (number of data sets included in the interval) and average learning method's performance for the three preprocessing techniques. Therefore, we can evaluate the performance of C4.5 and PART when using the oversampling and undersampling approaches by means of observing differences in the covered data sets by the obtained rules. These differences will provide information about the behavior of the three considered preprocessing approaches for C4.5 and PART.

This paper is organized as follows: first, Sect. 2 presents the problem of imbalanced data sets, describing its features, the preprocessing methods used, and the metric we have employed in this context. Next, Sect. 3 introduces the data complexity metrics we have used along with recent studies in the topic. In Sect. 4 the background on the use of data complexity for imbalanced data and the experimental framework used in this study are presented. In Sect. 5 the analyses of the methodology used and the experimental results are performed. Section 6 summarizes and concludes the work. Appendix 1 contains the figures with the intervals extracted in the study. Appendix 2 depicts the tables with the average results obtained for each data set in the study.

2 Imbalanced data sets in classification

In this section, the problem of imbalanced data sets in Sect. 2.1 is introduced first. The SMOTE and SMOTE-ENN are described in Sect. 2.2. The EUSCHC method is described in Sect. 2.3. Finally, Sect. 2.4 presents the evaluation metrics for this kind of classification problems.

2.1 The problem of imbalanced data sets

In the classification problem field, the scenario of imbalanced data sets appears when the number of examples that represent the different classes are very different among them (Chawla et al. 2004). We focus on the binary-class imbalanced data sets, where there is only one positive (minority) and one negative (majority) class. In this work we consider the IR (Orriols-Puig and Bernadó-Mansilla 2008), defined as the number of negative class instances divided by the number of positive class instances. The IR can be used to organize the different data sets according to their degree of imbalance.

Most of the learning algorithms aim to obtain a model with a high prediction accuracy and a good generalization capability. However, this inductive bias towards such a model supposes a serious challenge with the classification of imbalanced data (Sun et al. 2009). First, if the search process is guided by the standard accuracy rate, it benefits the covering of the majority examples; second, classification rules that predict the positive class are often highly specialized and thus their coverage is very low; hence they are discarded in favor of more general rules, i.e., those that predict the negative class. Furthermore, it is not easy to distinguish between noise examples and minority class

examples and they can be completely ignored by the classifier.

In recent years regarding real world domains the importance of the imbalance learning problem grows since it is a recurring problem in many applications, such as remote-sensing (Williams et al. 2009), pollution detection (Lu and Wang 2008) and especially medical diagnosis (Kilic et al. 2007; Mazurowki et al. 2008; Celebi et al. 2007; Peng and King 2008).

For this reason, a large number of approaches have been previously proposed to deal with the class imbalance problem. These approaches can be categorized into two groups: the internal approaches that create new algorithms or modify existing ones to take the class imbalance problem into consideration (Barandela et al. 2003; Diamantini an Potena 2009) and external approaches that preprocess the data in order to diminish the effect cause by their class imbalance (Fernández et al. 2008; Drown et al. 2009; Tang et al. 2009). Furthermore, cost-sensitive learning solutions incorporating both the data and algorithmic level approaches assume higher misclassification costs with samples in the minority class and seek to minimize the high cost errors (Domingos 1999; Sun et al. 2007; Zhou and Liu 2006).

The great advantage of the external approaches is that they are more versatile, since their use is independent of the classifier selected. Furthermore, we may preprocess all data sets before-hand in order to use them to train different classifiers. In this manner, the computation time needed to prepare the data is only used once.

2.2 Oversampling approaches: the SMOTE and SMOTE-ENN algorithms

As mentioned before, applying a preprocessing step in order to balance the class distribution is a positive solution to the imbalance data set problem (Batista et al. 2004). Specifically, in this work we have chosen an over-sampling method which is a reference in this area: the SMOTE algorithm (Chawla et al. 2002), and a variant called SMOTE-ENN (Batista et al. 2004).

In SMOTE the minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of oversampling required, neighbors from the k-nearest neighbors are randomly chosen. This process is illustrated in Fig. 1, where x_i is the selected point, x_{i1} to x_{i4} are some selected nearest neighbors and r_1 to r_4 the synthetic data points created by the randomized interpolation. The implementation employed in this work uses the euclidean distance, and balances both classes to the 50% distribution.

Synthetic samples are generated in the following way: take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general. An example is detailed in Fig. 2.

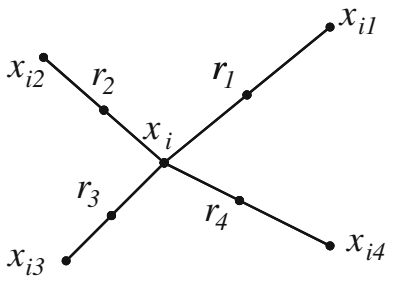

Fig. 1 An illustration on how to create the synthetic data points in the SMOTE algorithm

```
Consider a sample (6,4) and let (4,3) be its nearest neighbour.
(6,4) is the sample for which k-nearest neighbours are being
identified (4,3) is one of its k-nearest neighbours.
Let: f1_1 = 6 f2_1 = 4, f2_1 - f1_1 = -2
f1_2 = 4 f2_2 = 3, f2_2 - f1_2 = -1
The new samples will be generated as
f1',f2') = (6,4) + rand(0-1) * (-2,-1)
rand(0-1) generates a random number between 0 and 1.
```

Fig. 2 Example of the SMOTE application

In short, its main idea is to form new minority class examples by interpolating between several minority class examples that lie together. Thus, the overfitting problem is avoided and causes the decision boundaries for the minority class to spread further into the majority class space.

On the other hand, class clusters could not be well defined since some minority class examples might be invading the majority class space. This situation can occur when interpolating minority class examples can expand the minority class clusters, introducing artificial minority class examples too deeply in the majority class space. Inducing a classifier under such a situation can lead to overfitting. Batista et al. proposed to apply ENN to the over-sampled training set as a data cleaning method. ENN removes any example whose class label differs from the class of at least two of its three nearest neighbors. Thus, any example that is misclassified by its three nearest neighbors is removed from the training set. We refer to this technique as SMOTE-ENN.

2.3 Undersampling approach: the EUSCHC algorithm

Instead of creating new examples of the minority class, the undersampling approach selects a subset of the examples which represents the initial problem better, and avoids the

Table 1 Confusion matrix for a two-class problem

	Positive prediction	Negative prediction
Positive class	True positive (TP)	False negative (FN)
Negative class	False positive (FP)	True negative (TN)

bias to the minority class by removing redundant examples. This approach has also the advantage of creating a reduced set of examples to the induction process, making it less costly.

The search for the optimal subset of examples which affect the learning method's performance the less can be considered as a search problem in which evolutionary algorithms can be applied. In this work the EUSCHC technique (García and Herrera 2009a) is considered. EUSCHC is an evolutionary undersampling technique, which removes redundant noisy and redundant examples.

EUSCHC uses the well-known CHC evolutionary algorithm (Eshelman 1991) as a base for the selection of the subset of the examples, considering a binary codification for the subset membership. EUSCHC can also use any performance measure as fitness, weighting positively the correctly classified examples which are outside the selected subset.

2.4 Evaluation in imbalanced domains

The measures of the quality of classification are built from a confusion matrix (shown in Table 1) which records correctly and incorrectly recognized examples for each class. The most used empirical measure, accuracy (1), does not distinguish between the number of correct labels of different classes, which in the framework of imbalanced problems may lead to erroneous conclusions. As a classical example, if the ratio of imbalance presented in the data is 10:100, i.e., there is ten positive instance versus ninety negatives, then a classifier that obtains an accuracy rate of 90% is not truly accurate if it does not correctly cover the single minority class instance

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{1}$$

One appropriate metric that could be used to measure the performance of classification over imbalanced data sets is the Receiver Operating Characteristic (ROC) graphics (Bradley 1997). In these graphics the tradeoff between the benefits and costs can be visualized. They show that any classifier cannot increase the number of true positives without also increasing the false positives. The Area Under the ROC Curve (AUC) (Huang and Ling 2005) corresponds to the probability of correctly identifying which of the two stimuli is noise and which is signal plus noise. AUC provides a single-number summary for the performance of learning algorithms.

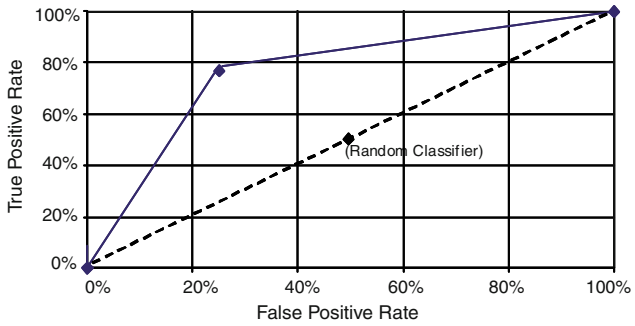

Fig. 3 Example of a ROC plot. Two classifiers are represented: the *solid line* is a good performing classifier whereas the *dashed line* represents a random classifier

The way to build the ROC space is to plot on a two-dimensional chart the true positive rate (Y axis) against the false positive rate (X axis) as shown in Fig. 3. The points (0, 0) and (1, 1) are trivial classifiers in which the output class is always predicted as negative and positive, respectively, while the point (0, 1) represents perfect classification. To compute the AUC we just need to obtain the area of the graphic as

$$\mathrm{AUC} = \frac{1 + \mathrm{TP}_{\mathrm{rate}} - \mathrm{FP}_{\mathrm{rate}}}{2}, \tag{2}$$

where $\mathrm{TP}_{\mathrm{rate}}$ and $\mathrm{FP}_{\mathrm{rate}}$ are the percentage of correctly and wrongly classified cases belonging to the positive class, respectively.

3 Data complexity

In this section we first present a short review on recent studies in data complexity in Sect. 3.1. The data complexity measures considered in this paper are described in Sect. 3.2.

3.1 Recent studies on data complexity measures

One direct approach to deal with data complexity is to obtain indicators about it by means of some measures. In particular, Ho and Basu (2002) proposed some complexity measures for binary classification problems, gathering metrics of three types: overlaps in feature values from different classes; separability of classes and measures of geometry, topology, and density of manifolds. Shortly after, Singh (2003) offered a review of data complexity measures of different nature [from Bayes error-based to nonparametric methods of Ho and Basu (2002)] and proposed two new ones.

Table 2 Data complexity measures names and acronyms proposed by Ho and Basu

Type	Id.	Description
Measures of overlaps in feature values from different classes	F1	Maximum Fisher's discriminant ratio
	F2	Error rate of linear classifier by linear programming
	F3	Maximum (individual) feature efficiency
Measures of separability of classes	L1	Minimized sum of error distance by linear programming
	L2	Error rate of linear classifier by linear programming
	N1	Fraction of points on class boundary
	N2	Ratio of average intra/inter class NN distance Ms Cercanos intra/inter-clases
	N3	Error rate of 1NN classifier
Measures of geometry, topology and density of manifolds	L3	Nonlinearity of linear classifier by linear programming
	N4	Non-linearity of 1NN classifier
	T1	Fraction of points with associated adherence subsets retained
	T2	Average number of points per dimension

These two studies, especially (Ho and Basu 2002), have been widely used afterwards. In the field of classification, we can find recent works using the measures of Ho and Basu. Bernadó-Mansilla and Ho (2005) investigated the domain of competence of XCS by means of a methodology that characterizes the complexity of a classification problem by a set of geometrical descriptors. Li et al. (2005) analyzed some omnivariate decision trees using the measure of complexity based in data density. Baumgartner and Somorjai (2006) defined specific measures for regularized linear classifiers. Sánchez et al. (2007) analyzed the effect of the data complexity in the nearest neighbors classifier, while García et al. (2009c) studied the relationship of the Fisher's discriminant ratio with respect to an evolutionary instance selection method in the classification task.

Focusing on how some data complexity measures affect the practical accuracy of these classification algorithms, they show which data complexity measures appear to better describe the behavior of the classifiers. More recently, Luengo and Herrera (2010) analyzed the domains of competence of a Fuzzy Rule-Based Classification System with respect to eight data complexity measures from Ho and Basu (2002). In the latter work, descriptive rules of good and bad behaviors of the method were defined based on the characteristics of the data sets characterized by the data complexity measures.

On the other hand, there exist proposals which do not use the measures of Ho and Basu with respect to a classification method directly, but taking into account a preprocessing technique. Dong and Kothari (2003) proposed a feature selection algorithm based on a complexity measure defined by Ho and Basu. Mollineda et al. (2005) extend some of Ho and Basu's measure definitions for problems with more than two classes. They analyzed these generalized measures in two classic Prototype Selection algorithms and remarked that Fisher's discriminant ratio is the most effective for Prototype Selection. Considering the opposite case, Kim and Oommen (2009) analyzed how to use prototype selection in order to decrease the computation time of several data complexity measures, without severely affecting the outcome with respect to the complete data set.

3.2 Data complexity measures

As we have mentioned, data complexity measures are a set of metrics that quantify characteristics which imply some difficulty to the classification task. In our analysis we will initially consider the 12 measures used in Ho and Basu (2002) for standard classification in the imbalanced framework used in this paper. The 12 measures are summarized in Table 2.

In our analysis, only F1, N4, and L3 measures of the 12 presented in Table 2 proved to be informative following the methodology described in Sect. 5.1 and observed in the experimental results in Sect. 5.2. The description of these three measures is included next.

F1: maximum Fisher's discriminant ratio. Fisher's discriminant ratio for one feature dimension is defined as

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where μ_1, μ_2, σ_1^2, σ_2^2 are the means and variances of the two classes, respectively, in that feature dimension. We compute f for each feature and take the maximum as measure F1. For a multidimensional problem, not all features have to contribute to class discrimination. The problem is easy as long as there exists one discriminating feature. Therefore, we can just take the maximum f over all feature dimensions in discussing class separability.

L3: nonlinearity of linear classifier by LP. Hoekstra and Duin (1996) proposed a measure for the nonlinearity of a

classifier with respect to a given data set. Given a training set, the method first creates a test set by linear interpolation (with random coefficients) between randomly drawn pairs of points from the same class. Then the error rate of the classifier (trained by the given training set) on this test set is measured. Here, we use such a nonlinearity measure for the linear classifier defined for L1. In particular, we consider a Support Vector Machine with a linear Kernel, which acts as a linear discriminant in this case. This measure is sensitive to the smoothness of the classifier's decision boundary as well as the overlap of the convex hulls of the classes. For linear classifiers and linearly separable problems, it measures the alignment of the decision surface with the class boundary. It carries the effects of the training procedure in addition to those of the class separation.

N4: nonlinearity of 1NN classifier. This measure follows the same procedure described for L3. In the case of N4, error is calculated for a nearest neighbor classifier. This measure is for the alignment of the nearest-neighbor boundary with the shape of the gap or overlap between the convex hulls of the classes.

4 On the use of data complexity measures for imbalanced data

In this section we first present the imbalanced data considered in this study and the configuration used for the calculation of the data complexity measures in Sect. 4.1, then reasons which motivates their use are introduced in Sect. 4.2.

4.1 Data sets and configuration of the methods

In order to analyze the preprocessing of the SMOTE, SMOTE-ENN and EUSCHC methods, we have selected 44 data sets from UCI repository (Asuncion and Newman 2007). The data are summarized in Table 3, showing the number of examples (#Ex.), attributes (#Atts.), name of each class (minority and majority), class attribute distribution, IR and F1, N4, and L3 data complexity values associated.

For every binary data set generated, we computed the 12 data complexity measures of Ho and Basu (2002) over the complete data set before preprocessing and splitting the data. Table 3 contains the F1, N4, and L3 measures' values for each original data set, as they proved to be the informative ones in our study. This will provide us information about the nature of the complete data set before preprocessing and applying the validation scheme.

The calculation of the data complexity measures supports some variants. The particular details of the computation of the measures that we have followed are detailed next.

- The instances with missing values are discarded previously to the measures calculation.
- The measures calculation over the data sets is performed with the original values, without any kind of normalization.
- The distance function used for continuous values is the normalized Euclidean distance function, i.e., the distance of each attribute is normalized by its range.
- The distance function used for nominal values is the overlap distance function. That is, if two nominal attributes are equal, the distance between them is 0. Otherwise the distance is 1.

It is essential to maintain this configuration for all the data sets, as any change in it has proven to produce changes in the estimated complexity obtained. Therefore, alterations in the distance functions used, for example, can disturb the analysis done and the conclusions obtained from it.

In order to carry out the different experiments we consider a *5-folder cross-validation model*, i.e., 5 random partitions of data with a 20%, and the combination of 4 of them (80%) as training and the remaining one as test. For each data set we consider the average results of the five partitions.

Then, in order to reduce the effect of imbalance, we will employ the SMOTE and SMOTE-ENN preprocessing method for all our experiments balancing both classes to the 50% distribution in the training partition (Batista et al. 2004). EUSCHC aims at reducing the data in the training partition as much as possible while the performance in AUC is not decreased.

The C4.5 and PART algorithms were run using KEEL[1] software (Alcalá-Fdez et al. 2009) following the recommended parameter values given in this platform, which must also correspond to the ones given by the authors in the original papers:

- For C4.5 the minimum number of item-sets per leaf was set to 2, and a pruning step is applied for the final tree with a confidence level of 0.25.
- For PART the minimum number of item-sets per leaf was also set to 2, and a pruning step is applied for the final tree with a confidence level of 0.25 as well.

In Table 4 we have summarized the global average for training and test AUC and the corresponding standard deviation obtained by C4.5 with SMOTE, SMOTE-ENN and EUSCHC preprocessing. These two tables are mean to be used for further reference in the analysis of the behavior of the preprocessing techniques in the following sections. As a general comment, SMOTE and SMOTE-ENN produce C4.5 and PART have a better training adjustment, while EUSCHC allows C4.5 and PART to generalize

[1] http://keel.es.

Table 3 Summary description for imbalanced data sets

Data set	#Ex.	#Atts.	Class (min., maj.)	%Class (min.; maj.)	IR	F1	N4	L3
Glass1	214	9	(build-win-non oat-proc; remainder)	(35,51; 64,49)	1.82	0.1897	0.3084	0.5000
Ecoli0vs1	220	7	(im; cp)	(35,00; 65,00)	1.86	9.7520	0.0136	0.1182
Wisconsin	683	9	(malignant; benign)	(35,00; 65,00)	1.86	3.5680	0.0432	0.0066
Pima	768	8	(tested-positive; tested-negative)	(34,84; 66,16)	1.90	0.5760	0.2754	0.5000
Iris0	150	4	(Iris-Setosa; remainder)	(33,33; 66,67)	2.00	16.8200	0.0000	0.0000
Glass0	214	9	(build-win-oat-proc; remainder)	(32,71; 67,29)	2.06	0.6492	0.2009	0.5000
Yeast1	1484	8	(nuc; remainder)	(28,91; 71,09)	2.46	0.2422	0.3201	0.5000
Vehicle1	846	18	(Saab; remainder)	(28,37; 71,63)	2.52	0.3805	0.1761	0.2311
Vehicle2	846	18	(Bus; remainder)	(28,37; 71,63)	2.52	0.1691	0.3304	0.3682
Vehicle3	846	18	(Opel; remainder)	(28,37; 71,63)	2.52	0.1855	0.3747	0.3511
Haberman	306	3	(Die; Survive)	(27,42; 73,58)	2.68	0.1850	0.3431	0.4967
Glass0123vs456	214	9	(non-window glass; remainder)	(23,83; 76,17)	3.19	3.3240	0.0561	0.3294
Vehicle0	846	18	(Van; remainder)	(23,64; 76,36)	3.23	1.1240	0.1734	0.1219
Ecoli1	336	7	(im; remainder)	(22,92; 77,08)	3.36	2.6500	0.1265	0.5000
New-thyroid2	215	5	(hypo; remainder)	(16,89; 83,11)	4.92	3.5790	0.0233	0.2791
New-thyroid1	215	5	(hyper; remainder)	(16,28; 83,72)	5.14	3.5790	0.0209	0.2721
Ecoli2	336	7	(pp; remainder)	(15,48; 84,52)	5.46	1.8260	0.0685	0.5000
Segment0	2308	19	(brickface; remainder)	(14,26; 85,74)	6.01	1.7980	0.0358	0.5000
Glass6	214	9	(headlamps; remainder)	(13,55; 86,45)	6.38	2.3910	0.0537	0.5000
Yeast3	1484	8	(me3; remainder)	(10,98; 89,02)	8.11	2.7510	0.1122	0.5000
Ecoli3	336	7	(imU; remainder)	(10,88; 89,12)	8.19	1.5790	0.1652	0.5000
Page-blocks0	5472	10	(remainder; text)	(10,23; 89,77)	8.77	0.5087	0.2069	0.3332
Yeast2vs4	514	8	(cyt; me2)	(9,92; 90,08)	9.08	1.5790	0.1333	0.5000
Yeast05679vs4	528	8	(me2; mit, me3, exc, vac, erl)	(9,66; 90,34)	9.35	1.0510	0.2509	0.5000
Vowel0	988	13	(hid; remainder)	(9,01; 90,99)	10.10	2.4580	0.2034	0.5000
Glass016vs2	192	9	(ve-win-oat-proc; build-win-oat-proc, build-win-non oat-proc,headlamps)	(8,89; 91,11)	10.29	0.2692	0.2891	0.5000
Glass2	214	9	(Ve-win-oat-proc; remainder)	(8,78; 91,22)	10.39	0.3952	0.3364	0.5000
Ecoli4	336	7	(om; remainder)	(6,74; 93,26)	13.84	3.2470	0.0506	0.5000
Yeast1vs7	459	8	(nuc; vac)	(6,72; 93,28)	13.87	12.9700	0.0016	0.0019
Shuttle0vs4	1829	9	(Rad Flow; Bypass)	(6,72; 93,28)	13.87	0.3534	0.3137	0.5000
Glass4	214	9	(containers; remainder)	(6,07; 93,93)	15.47	1.4690	0.1285	0.5000
Page-blocks13vs2	472	10	(graphic; horiz.line, picture)	(5,93; 94,07)	15.85	1.5470	0.0540	0.0678
Abalone9vs18	731	8	(18; 9)	(5,65; 94,25)	16.68	0.6320	0.3324	0.5000
Glass016vs5	184	9	(tableware; build-win-oat-proc, build-win-non oat-proc, headlamps)	(4,89; 95,11)	19.44	1.8510	0.0788	0.5000
Shuttle2vs4	129	9	(Fpv Open; Bypass)	(4,65; 95,35)	20.50	12.1300	0.0155	0.0000
Yeast1458vs7	693	8	(vac; nuc, me2, me3, pox)	(4,33; 95,67)	22.10	0.1757	0.3752	0.5000
Glass5	214	9	(tableware; remainder)	(4,20; 95,80)	22.81	1.0190	0.0724	0.5000
Yeast2vs8	482	8	(pox; cyt)	(4,15; 95,85)	23.10	1.1420	0.2261	0.5000
Yeast4	1484	8	(me2; remainder)	(3,43; 96,57)	28.41	0.7412	0.2342	0.5000
Yeast1289vs7	947	8	(vac; nuc, cyt, pox, erl)	(3,17; 96,83)	30.56	0.3660	0.3627	0.5000
Yeast5	1484	8	(me1; remainder)	(2,96; 97,04)	32.78	4.1980	0.1216	0.5000
Ecoli0137vs26	281	7	(pp, imL; cp, im, imU, imS)	(2,49; 97,51)	39.15	1.9670	0.1701	0.5000
Yeast6	1484	8	(exc; remainder)	(2,49; 97,51)	39.15	2.3020	0.1157	0.5000
Abalone19	4174	8	(19; remainder)	(0,77; 99,23)	128.87	0.5295	0.4534	0.5000

Table 4 Global average Training and Test AUC for C4.5

	Global % AUC Training	Global % AUC Test
C4.5 with SMOTE preprocessing	0.9546 ± 0.0551	0.8217 ± 0.1375
C4.5 with SMOTE-ENN preprocessing	0.9438 ± 0.0635	0.8362 ± 0.1309
C4.5 with EUSCHC preprocessing	0.9241 ± 0.0859	0.8914 ± 0.1035

Table 5 Global average Training and Test AUC for PART

	Global % AUC Training	Global % AUC Test
PART with SMOTE preprocessing	0.9440 ± 0.0727	0.8298 ± 0.1368
PART with SMOTE-ENN preprocessing	0.9353 ± 0.0727	0.8372 ± 0.1313
PART with EUSCHC preprocessing	0.9172 ± 0.0796	0.8900 ± 0.0899

better. In Table 5 we present the same AUC averages for PART. The complete tables of results are shown in Appendix 2.

4.2 Motivation of the use of complexity measures

In Sect. 2, the necessity of the use of instance preprocessing in the framework of imbalanced data has been shown. The IR measure has been also used in order to classify the different imbalanced problems based on their imbalanced degree. However, we have observed empirically that this measure has not a clear relationship with the performance obtained with the preprocessing techniques. In this sense, Fig. 4 depicts the results for C4.5 in the case of preprocessing the 44 data sets with SMOTE and EUSCHC, sorting the data sets by their IR value. We can observe that the good and bad results of both learning methods with respect to the preprocessing are not related with the IR value, nor the improvements achieved with such preprocessing step.

Therefore, the use of the IR as a unique measure to identify the improvement of the preprocessing appears to be insufficient, and we need to consider other measures to characterize the good or bad behavior of the preprocessing, like the data complexity measures presented in Sect. 3.2.

To the best of our knowledge there is no analysis on the relationship of the data complexity and the application of preprocessing techniques for imbalanced data. García et al. (2008) built a bunch on synthetic data sets with a wide range of overlapping present in the two classes. Using this framework, the response of local and global learning methods (the *k*-NN classifier and several others, C4.5 among them) is studied when varying the IR and the overlapping between the class labels. Albeit they do not explicitly used the data complexity measures of Ho and Basu (2002), the class overlapping and IR can be considered as related to them. Results showed that the more represented class in overlapped regions tends to be better classified by methods based on global learning, while the less class represented in such regions tends to be better classified by local methods.

However, in García et al.'s (2008) study no preprocessing was performed. In this paper we will analyze the mentioned the undersampling and oversampling preprocessing approaches by means of the data complexity measures.

We emphasize that this study does not attempt to establish the best preprocessing method for a given

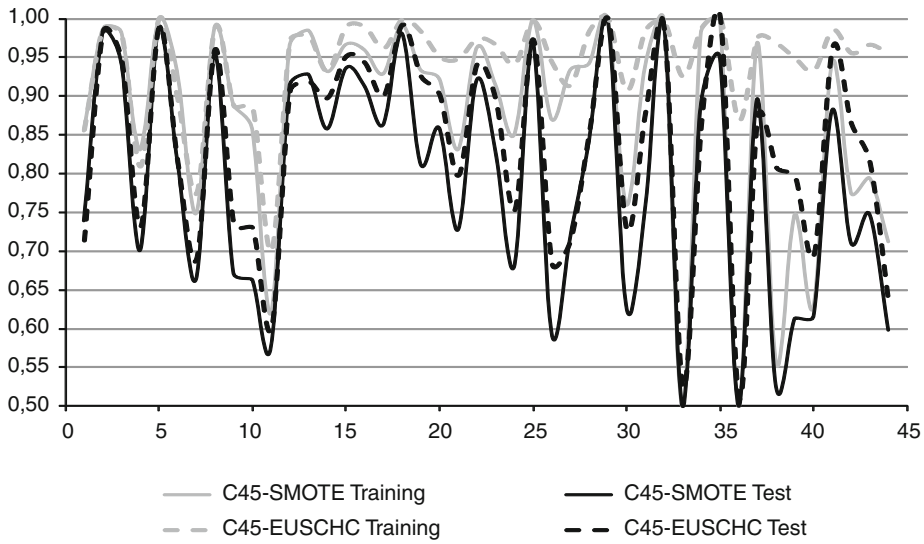

Fig. 4 C4.5 AUC in Training/Test sorted by IR

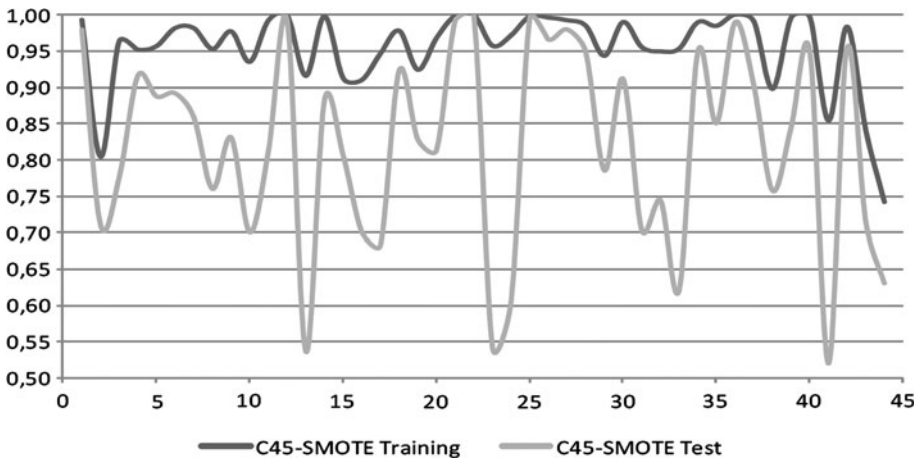

Fig. 5 C4.5 AUC with SMOTE in Training/Test sorted by F2

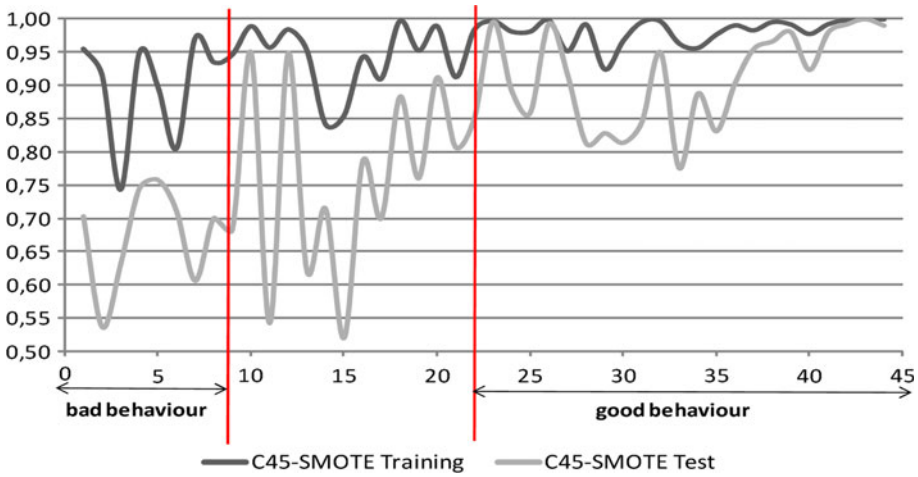

Fig. 6 C4.5 AUC with SMOTE in Training/Test sorted by F1

problem. This estimation problem has been already formalized as a new learning problem in the Meta-Learning approach (MetaL) (Brazdil et al. 2009). The MetaL approach faces two important drawbacks:

- How to represent an ML problem instance was tackled using diverse descriptors, e.g., number of examples, number of attributes, percentage of missing values, and landmarkers (Pfahringer et al. 2000). The difficulty is due to the fact the descriptors must take into account the example distribution, which is not easily achieved in most cases.
- A second difficulty concerns the selection of the ML problem instances. Kalousis (2002) indicates that the representativity of the problems and the perturbation induce strong biases in the Metal classifier.

Due to the several difficulties already studied in the specialized literature, the attempt to indicate the best preprocessing method is out of the scope of this paper.

Table 6 Significant intervals for C4.5 and PART

With SMOTE and SMOTE-ENN Interval	With EUSCHC Interval	Behavior
C4.5		
F1 $\geq$ 1.469	F1 $\geq$ 0.6492	Good
N4 $\leq$ 0.2069	N4 $\leq$ 0.2509	Good
L3 $\leq$ 0.3332	L3 $\leq$ 0.3332	Good
F1 $\leq$ 0.366	F1 $\leq$ 0.3534	Bad
N4 $\geq$ 0.2261	N4 $\geq$ 0.2754	Bad
PART		
F1 $\geq$ 1.469	F1 $\geq$ 0.632	Good
N4 $\leq$ 0.2069	N4 $\leq$ 0.2509	Good
L3 $\leq$ 0.3332	L3 $\leq$ 0.3332	Good
F1 $\leq$ 0.366	F1 $\leq$ 0.3534	Bad
N4 $\geq$ 0.2261	N4 $\geq$ 0.2754	Bad

5 Analysis of the influence of preprocessing in imbalanced data

In this study, our aim is to analyze the suitability of the use of data complexity measures to evaluate the behavior of SMOTE, SMOTE-ENN, and EUSCHC using C4.5 and PART in the scenario of imbalanced data sets.

Table 7 Rules with one metric obtained from the intervals for C4.5

Id.	Rule	Preprocess	Support	Avg. AUC Train	Train diff.	Avg. AUC Test	Test diff.
R1+	If F1 $\geq$ 1.469 then good behavior	SMOTE	52.27	0.9826	0.028	0.9103	0.0886
		SMOTE-ENN		0.9785	0.0347	0.9234	0.0872
	If F1 $\geq$ 0.6482 then good behavior	EUSCHC	65.91	0.9687	0.0515	0.9450	0.1632
R2+	If L3 $\leq$ 0.3332 then good behavior	SMOTE	27.27	0.9929	0.0383	0.9641	0.1424
		SMOTE-ENN		0.9876	0.0438	0.9610	0.1248
		EUSCHC		0.9877	0.0636	0.9688	0.0774
R3+	If N4 $\leq$ 0.2069 then good behavior	SMOTE	63.63	0.9823	0.0277	0.9077	0.086
		SMOTE-ENN		0.9756	0.0318	0.9196	0.0834
	If N4 $\leq$ 0.2509 then good behavior	EUSCHC	70.45	0.9692	0.0451	0.9460	0.0546
R1–	If F1 $\leq$ 0.366 then bad behavior	SMOTE	20.45	0.9021	–0.0525	0.6748	–0.1469
		SMOTE-ENN		0.8613	–0.0825	0.6762	–0.1600
	If F1 $\leq$ 0.3534 then bad behavior	EUSCHC	18.18	0.8186	–0.1055	0.7519	–0.1395
R2–	If N4 $\geq$ 0.2261 then bad behavior	SMOTE	36.36	0.9062	–0.0484	0.6712	–0.1505
		SMOTE-ENN		0.8881	–0.0557	0.6903	–0.1459
	If N4 $\geq$ 0.2754 then bad behavior	EUSCHC	29.55	0.8166	–0.1075	0.7613	–0.1301

Table 8 Rules with one metric obtained from the intervals for PART

Id.	Rule	Preprocess	Support	Avg. AUC Train	Train diff.	Avg. AUC Test	Test diff.
R1+	If F1 $\geq$ 1.469 then good behavior	SMOTE	52.27	0.9817	0.0377	0.9194	0.0896
		SMOTE-ENN		0.9764	0.0411	0.9259	0.0887
	If F1 $\geq$ 0.632 then good behavior	EUSCHC	68.18	0.9538	0.0366	0.9322	0.0422
R2+	If L3 $\leq$ 0.3332 then good behavior	SMOTE	27.27	0.9932	0.0492	0.9646	0.1348
		SMOTE-ENN		0.9857	0.0504	0.9687	0.1315
		EUSCHC		0.9716	0.0544	0.9514	0.0614
R3+	If N4 $\leq$ 0.2069 then good behavior	SMOTE	63.63	0.9805	0.0365	0.9162	0.0864
		SMOTE-ENN		0.9736	0.0383	0.9203	0.0831
	If N4 $\leq$ 0.2509 then good behavior	EUSCHC	70.45	0.9564	0.0392	0.9349	0.0449
R1–	If F1 $\leq$ 0.366 then bad behavior	SMOTE	20.45	0.8637	–0.0803	0.6687	–0.1611
		SMOTE-ENN		0.8364	–0.0989	0.6618	–0.1754
	If F1 $\leq$ 0.3534 then bad behavior	EUSCHC	18.18	0.7914	–0.1258	0.7459	–0.1441
R2–	If N4 $\geq$ 0.2261 then bad behavior	SMOTE	36.36	0.8801	–0.0639	0.6788	–0.1510
		SMOTE-ENN		0.8684	–0.0669	0.6917	–0.1455
	If N4 $\geq$ 0.2754 then bad behavior	EUSCHC	29.55	0.8236	–0.0936	0.7831	–0.1069

In the remaining of this section, we will first show the methodology followed in this study in Sect. 5.1. Next the empirical study for both C4.5 and PART in imbalanced data sets with data complexity measures is shown in Sect. 5.2. Finally, in Sect. 5.3 the collective evaluation of the set of rules for both learning models is carried out.

5.1 Methodology

In order to characterize the results of C4.5 and PART when using undersampling and oversampling preprocessing in imbalanced data, we will follow the methodology proposed in (Luengo and Herrera 2010) for characterizing the performance of a learning method in standard classification, which is briefly described next.

We consider intervals of data complexity measures' values in which C4.5 and PART perform good or bad, calculated for every data set of Sect. 4.1.

- We understand for *good behavior* an average high test AUC in the interval (at least 0.8 approximately), as well as the absence of over-fitting (less than a 0.1 difference in Training and Test AUC approximately).
- By *bad behavior* we refer to the presence of over-fitting and/or average low test AUC in the interval.

Table 9 Data sets sorted by F1 covered by the R1+ and R1– rules

Data set	F1	C4.5, PART with SMOTE, SMOTE-ENN	C4.5 with EUSCHC	PART with EUSCHC
Vehicle2	0.1691	R1- bad behavior	R1- bad behavior	R1- bad behavior
Yeast1458vs7	0.1757			
Haberman	0.1850			
Vehicle3	0.1855			
Glass1	0.1897			
Yeast1	0.2422			
Glass016vs2	0.2692			
Shuttle0vs4	0.3534			
Yeast1289vs7	0.3660			
Vehicle1	0.3805			
Glass2	0.3952			
Page-blocks0	0.5087			
Abalone19	0.5295			
Pima	0.5760			
Abalone9vs18	0.6320			
Glass0	0.6492			
Yeast4	0.7412			
Glass5	1.0190			
Yeast05679vs4	1.0510			
Vehicle0	1.1240			
Yeast2vs8	1.1420			
Glass4	1.4690	R1+ good behavior	R1+ good behavior	R1+ good behavior
Page-blocks13vs2	1.5470			
Ecoli3	1.5790			
Yeast2vs4	1.5790			
Segment0	1.7980			
Ecoli2	1.8260			
Glass016vs5	1.8510			
Ecoli0137vs26	1.9670			
Yeast6	2.3020			
Glass6	2.3910			
Vowel0	2.4580			
Ecoli1	2.6500			
Yeast3	2.7510			
Ecoli4	3.2470			
Glass0123vs456	3.3240			
Wisconsin	3.5680			
New-thyroid2	3.5790			
New-thyroid1	3.5790			
Yeast5	4.1980			
Ecoli0vs1	9.7520			
Shuttle2vs4	12.1300			
Yeast1vs7	12.9700			
Iris0	16.8200			

Table 10 Data sets sorted by N4 covered by the R3+ and R2– rules

Data set	N4	C4.5, PART with SMOTE, SMOTE-ENN	C4.5 with EUSCHC	PART with EUSCHC
Iris0	0.0000	R3+ good behavior	R3+ good behavior	R3+ good behavior
Yeast1vs7	0.0016			
Ecoli0vs1	0.0136			
Shuttle2vs4	0.0155			
New-thyroid1	0.0209			
New-thyroid2	0.0233			
Segment0	0.0358			
Wisconsin	0.0432			
Ecoli4	0.0506			
Glass6	0.0537			
Page-blocks13vs2	0.0540			
Glass0123vs456	0.0561			
Ecoli2	0.0685			
Glass5	0.0724			
Glass016vs5	0.0788			
Yeast3	0.1122			
Yeast6	0.1157			
Yeast5	0.1216			
Ecoli1	0.1265			
Glass4	0.1285			
Yeast2vs4	0.1333			
Ecoli3	0.1652			
Ecoli0137vs26	0.1701			
Vehicle0	0.1734			
Vehicle1	0.1761			
Glass0	0.2009			
Vowel0	0.2034			
Page-blocks0	0.2069			
Yeast2vs8	0.2261	R2- bad behavior	R2- bad behavior	R2- bad behavior
Yeast4	0.2342			
Yeast05679vs4	0.2509			
Pima	0.2754			
Glass016vs2	0.2891			
Glass1	0.3084			
Shuttle0vs4	0.3137			
Yeast1	0.3201			
Vehicle2	0.3304			
Abalone9vs18	0.3324			
Glass2	0.3364			
Haberman	0.3431			
Yeast1289vs7	0.3627			
Vehicle3	0.3747			
Yeast1458vs7	0.3752			
Abalone19	0.4534			

The intervals are extracted by means of a particular graphic representation of the AUC results for C4.5 or PART considering the three preprocessing methods. The AUC results for the preprocessed data sets are arranged equidistantly in an ordered series, sorting them by one of the data complexity measures computed over the original data sets. Therefore, the *X* axis contains the data sets with a constant separation (and not related with the values of the considered data complexity), and the *Y* axis depicts the AUC obtained both in Training and Test for the particular data set. The reason to use this constant separation is to give each data set the same space in the graphic representation.

For those measures where we can find different *ad-hoc* intervals which present *good* or *bad behavior* of C4.5 or PART, we use a vertical line to delimit the interval of the region of interest.

Following the process described, not every data complexity measure can be used in order to select an interval of good or bad behavior of the methods. Figure 5 depicts an example in which no interval could be extracted for the F2 measure. Figure 6 shows an example in which both good and bad behavior intervals could be extracted, indicated by vertical lines for the F1 measure using the same preprocessing technique.

As mentioned in Sect. 3.2, only F1, N4 and L3 data complexity measures can be used to extract significative intervals with enough support following this methodology.

Table 11 Data sets sorted by L3 covered by the R2+ rule

Data set	L3	C4.5, PART with SMOTE, SMOTE-ENN	C4.5 with EUSCHC	PART with EUSCHC
Iris0	0.0000	R2+ good behavior	R2+ good behavior	R2+ good behavior
Yeast1vs7	0.0019			
Wisconsin	0.0066			
Page-blocks13vs2	0.0678			
Ecoli0vs1	0.1182			
Vehicle0	0.1219			
Vehicle1	0.2311			
New-thyroid1	0.2721			
New-thyroid2	0.2791			
Glass0123vs456	0.3294			
Page-blocks0	0.3332			
Vehicle3	0.3511			
Vehicle2	0.3682			
Haberman	0.4967			
Yeast1458vs7	0.5000			
Glass1	0.5000			
Yeast1	0.5000			
Glass016vs2	0.5000			
Shuttle0vs4	0.5000			
Yeast1289vs7	0.5000			
Glass2	0.5000			
Abalone19	0.5000			
Pima	0.5000			
Abalone9vs18	0.5000			
Glass0	0.5000			
Yeast4	0.5000			
Glass5	0.5000			
Yeast05679vs4	0.5000			
Yeast2vs8	0.5000			
Glass4	0.5000			
Ecoli3	0.5000			
Yeast2vs4	0.5000			
Segment0	0.5000			
Ecoli2	0.5000			
Glass016vs5	0.5000			
Ecoli0137vs26	0.5000			
Yeast6	0.5000			
Glass6	0.5000			
Vowel0	0.5000			
Ecoli1	0.5000			
Yeast3	0.5000			
Ecoli4	0.5000			
Yeast5	0.5000			

Table 12 Disjunction Rules from all simple rules for C4.5

Id.	Rule	Preprocess	Support (%)	Avg. AUC Train	Train diff.	Avg. AUC Test	Test diff.
PRD-S	If R1+ or R2+ R3+ then good behavior	SMOTE	63.63	0.9823	0.0277	0.9077	0.0861
PRD-S-ENN		SMOTE-ENN		0.9756	0.0318	0.9196	0.0834
PRD-EUS		EUSCHC	70.45	0.9692	0.0451	0.9460	0.0546
NRD-S	If R1– or R2– then bad behavior	SMOTE	36.36	0.9062	–0.0484	0.6712	–0.1504
NRD-S-ENN		SMOTE-ENN		0.8881	–0.0557	0.6903	–0.1459
NRD∧¬PRD-EUS		EUSCHC	29.55	0.8166	–0.1075	0.7613	–0.1301

Table 13 Disjunction Rules from all simple rules for PART

Id.	Rule	Preprocess	Support (%)	Avg. AUC Train	Train diff.	Avg. AUC Test	Test diff.
PRD-S	If R1+ or R2+ R3+ then good behavior	SMOTE	63.63	0.9805	0.0365	0.9162	0.0864
PRD-S-ENN		SMOTE-ENN		0.9736	0.0383	0.9203	0.0831
PRD-EUS		EUSCHC	72.73	0.9549	0.0377	0.9337	0.0437
NRD-S	If R1– or R2– then bad behavior	SMOTE	36.36	0.8801	–0.0639	0.6788	–0.1510
NRD-S-ENN		SMOTE-ENN		0.8684	–0.0669	0.6917	–0.1455
NRD∧¬PRD-EUS		EUSCHC	27.27	0.8167	–0.1005	0.7736	–0.1164

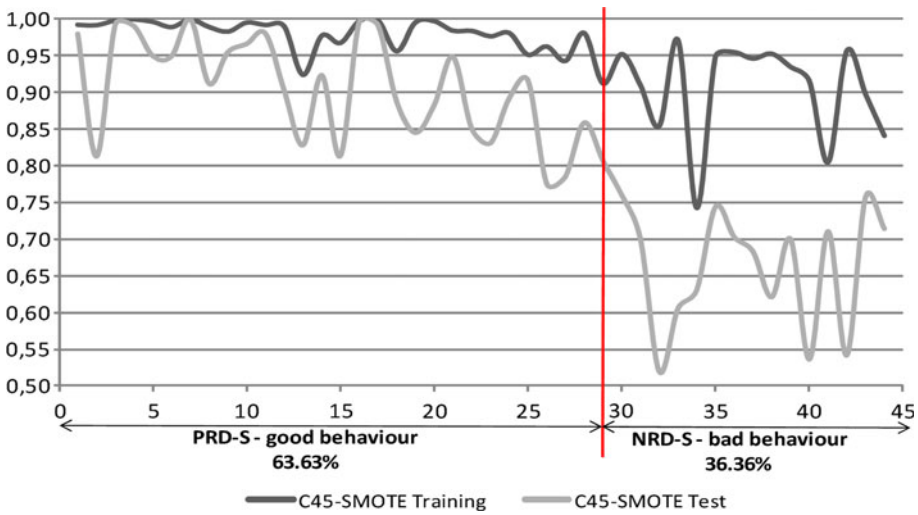

Fig. 7 Three blocks representation for C4.5 with SMOTE

Our objective is to analyze the data sets covered by a good or bad behavior interval considering the different preprocessing methods. These data sets will be characterized as those which the preprocessing technique used allows C4.5 and PART to obtain good or bad results.

5.2 Single intervals extraction

As we have previously indicated, only the F1, N4 and L3 measures offered significant intervals following the process described in Sect. 5.1. In Appendix 1, Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,

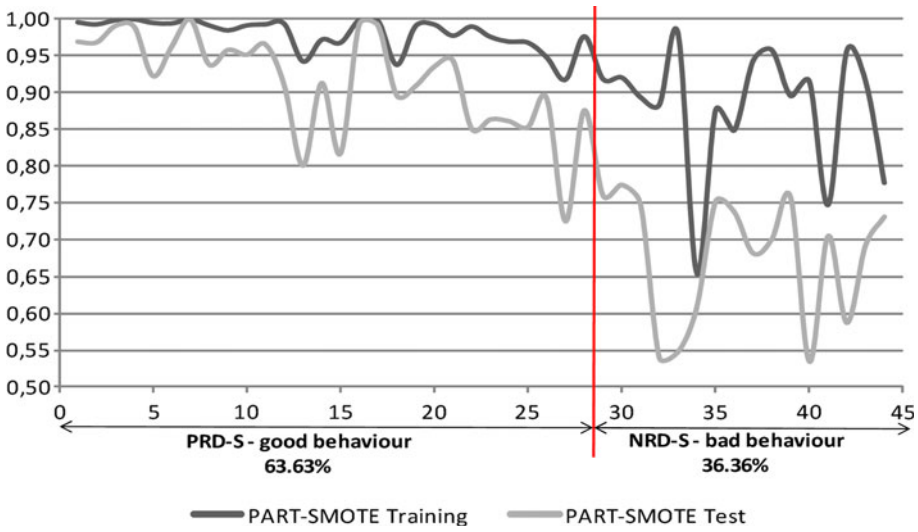

Fig. 8 Three blocks representation for PART with SMOTE

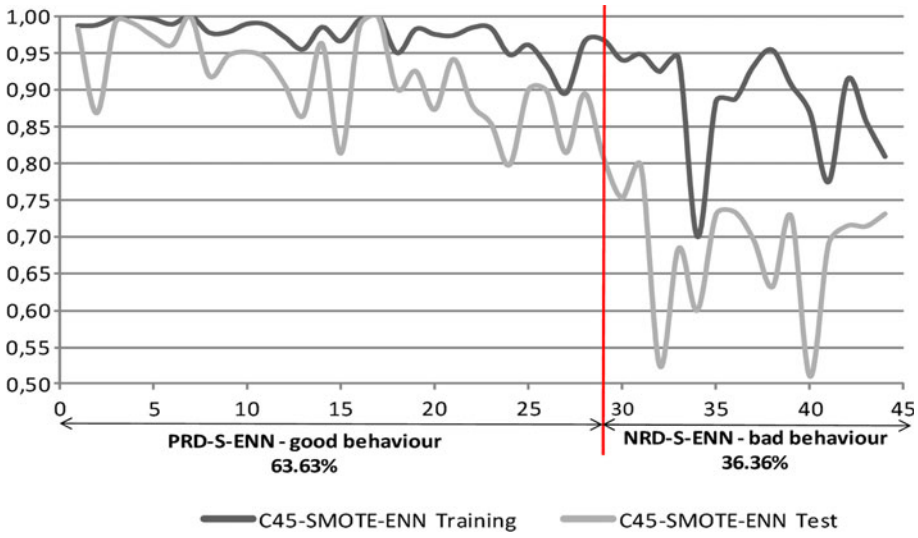

Fig. 9 Three blocks representation for C4.5 with SMOTE-ENN

26, 27, 28, 29, 30 depict the results for C4.5 and PART considering these three data complexity measures.

In Table 6 we have summarized the intervals found ad-hoc from the aforementioned figures for C4.5 and PART. The intervals are the same for C4.5 and PART. The intervals obtained for SMOTE and SMOTE-ENN are also always the same, therefore we will not find differences between the two versions of SMOTE considered, except in the average AUC, from now on.

All the extracted intervals can be translated into rules, using them as the antecedents of the rules. In Table 7 we have summarized the rules derived from the individual intervals for C4.5 and in Table 8 we show the equivalent rules for PART. Both tables are organized with the following columns:

- The first column corresponds to the identifier of the rule for further references.
- The "Rule" column presents the rule itself.
- The third column shows the type of preprocessing carried out.
- The fourth column "Support" presents the percentage of data sets which verifies the antecedent of the rule.
- The column "% Training" shows the average AUC in training of all the data sets covered by the rule.
- The column "Training diff." contains the difference between the training AUC of the rule and the global training AUC across all 44 data sets showed in Tables 4 and 5 for the preprocessing case of the row (SMOTE, SMOTE-ENN or EUSCHC).
- The column "% Test" shows the average AUC in test of all the data sets covered by the rule.

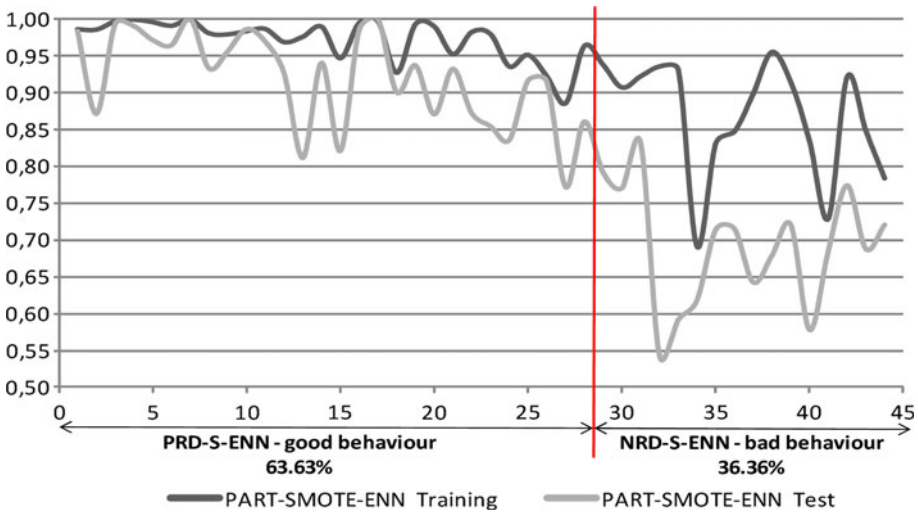

Fig. 10 Three blocks representation for PART with SMOTE-ENN

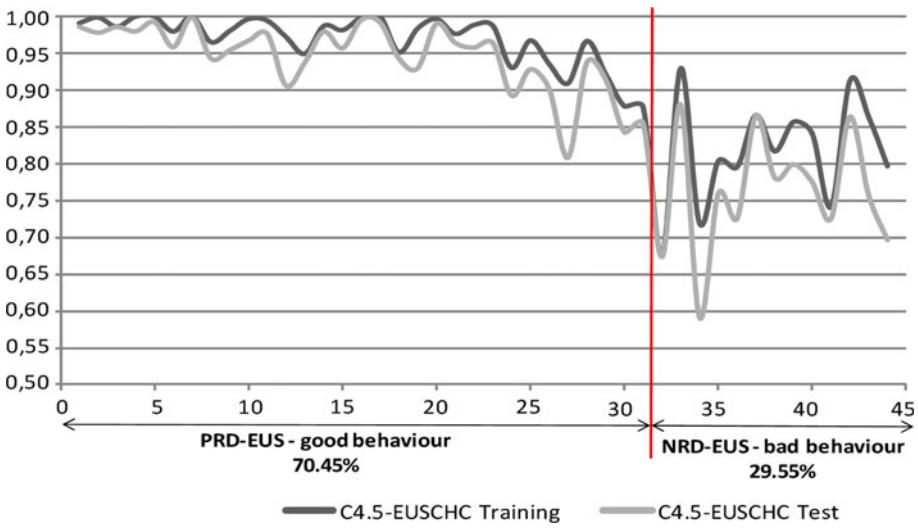

Fig. 11 Three blocks representation for C4.5 with EUSCHC

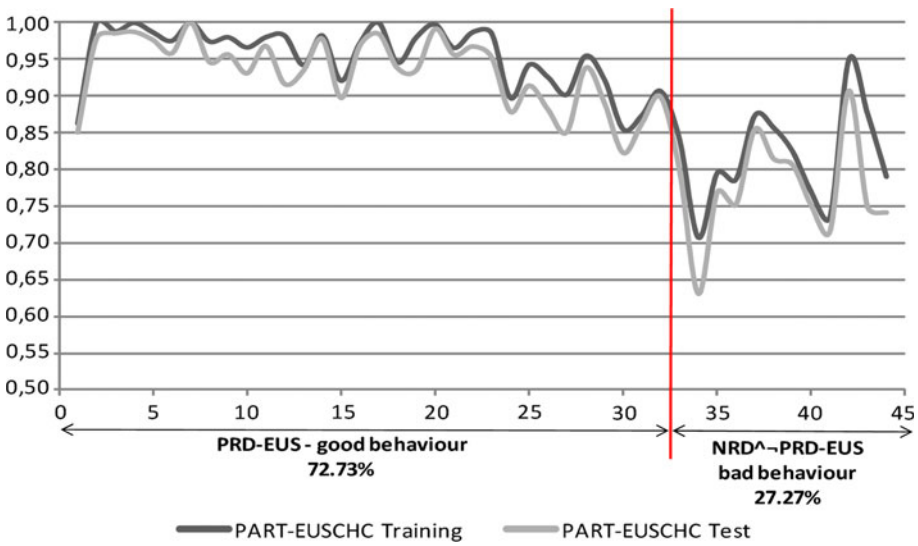

Fig. 12 Three blocks representation for PART with EUSCHC

Table 14 Data sets covered by the PRD and NRD rules

Data set	C4.5, PART with SMOTE, SMOTE-ENN	C4.5 with EUSCHC	PART with EUSCHC
Ecoli0vs1	PRD good behavior	PRD good behavior	PRD good behavior
Glass016vs5			
Segment0			
Iris0			
Vowel0			
Vehicle2			
shuttle0vs4			
Vehicle0			
Wisconsin			
New-Thyroid2			
New-Thyroid1			
Glass0123vs456			
Yeast6			
Yeast5			
Ecoli0137vs26			
Page-Blocks13vs4			
shuttle2vs4			
Yeast3			
Glass6			
Glass5			
Page-Blocks0			
Glass4			
Ecoli4			
Ecoli3			
Ecoli2			
Ecoli1			
Glass0			
Yeast2vs4			
Yeast2vs8	NRD bad behavior	NRD bad behavior	NRD∧¬PRD bad behavior
Yeast05679vs4			
Yeast4			
Abalone19			
Glass016vs2			
Haberman			
Vehicle3			
Vehicle1			
Yeast1289vs7			
Abalone9-18			
yeastB1vs7			
Yeast1458vs7			
Yeast2			
Glass2			
Glass1			
Pima			

- The column "Test diff." contains the difference between the test AUC of the rule and the global test AUC across all 44 data sets showed in Tables 4 and 5 for the preprocessing case of the row (SMOTE, SMOTE-ENN or EUSCHC).

The positive rules (denoted with a "+" symbol in their identifier) always show a positive difference with the global average, both in training and test AUC. The negative ones (with a "–" symbol in their identifier) verify the opposite case. The support of the rules shows us that we can characterize a wide range of data sets and obtain significant differences in AUC both inf with and without preprocessing cases.

In Tables 9, 10 and 11 the specific data sets arranged by the F1, N4, and L3 measures, respectively, are depicted. The data sets covered by each rule is indicated by means of the adjacent columns, considering the three different support cases obtained from the rules: SMOTE and SMOTE-ENN for both C4.5 and PART; EUSCHC for C4.5, and EUSCHC for PART.

If we compare the equivalent rules and the covered data sets in the different cases of SMOTE, SMOTE-ENN and EUSCHC preprocessing we can observe the following:

- With the use of EUSCHC the support of R1+ and R3+ rules with this preprocessing method is wider than the SMOTE–based approaches.
- The support of the R1– and R2– is also smaller for EUSCHC, as less data sets can be identified as bad for C4.5 or PART. These differences with respect to SMOTE and SMOTE-ENN has been properly characterized by the rules.
- The R2+ rules is invariant with respect to the preprocessing. It represents the good data sets for

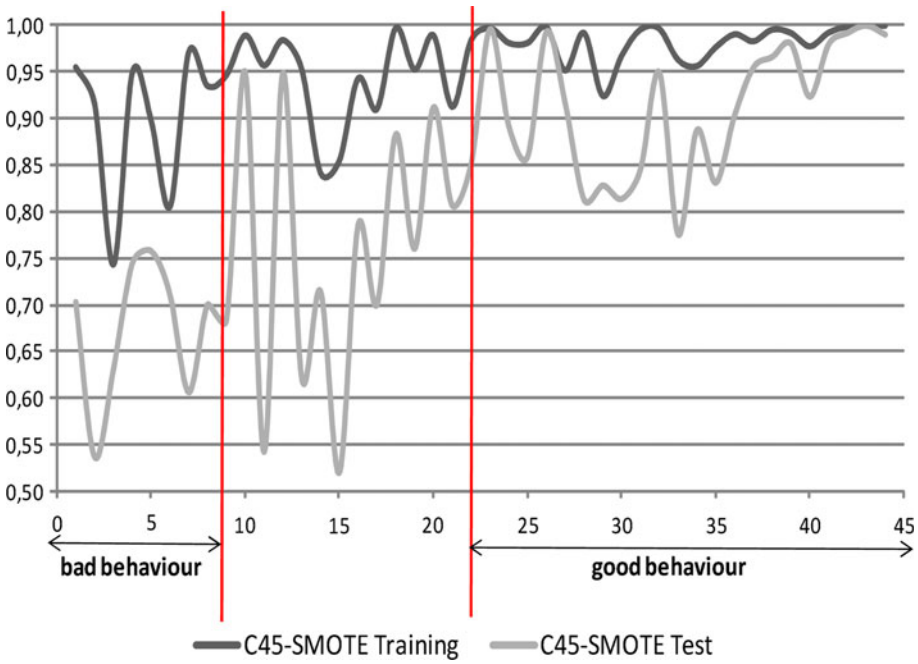

Fig. 13 C4.5 with SMOTE AUC in Training/Test sorted by F1

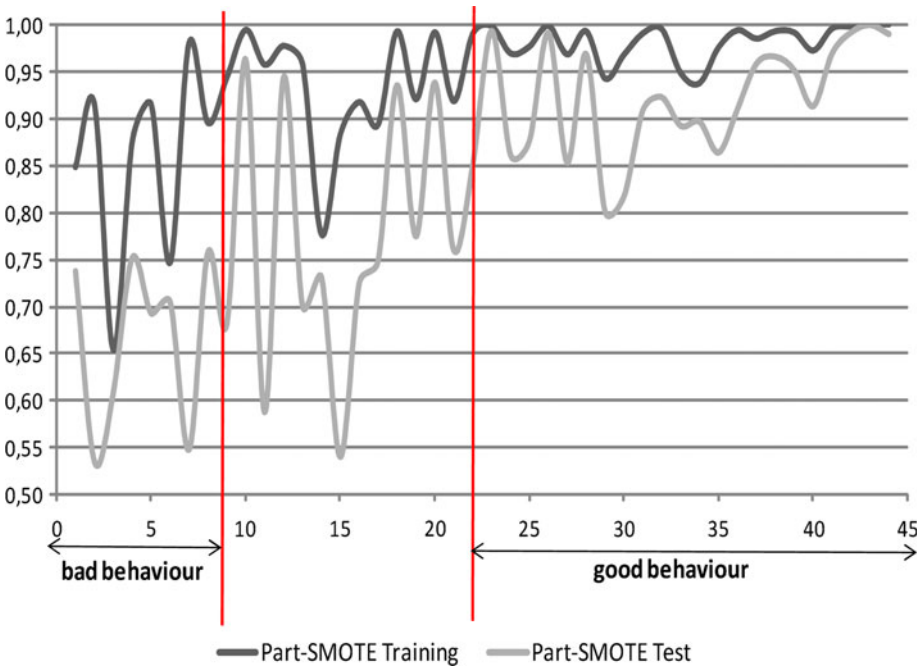

Fig. 14 PART with SMOTE AUC in Training/Test sorted by F1

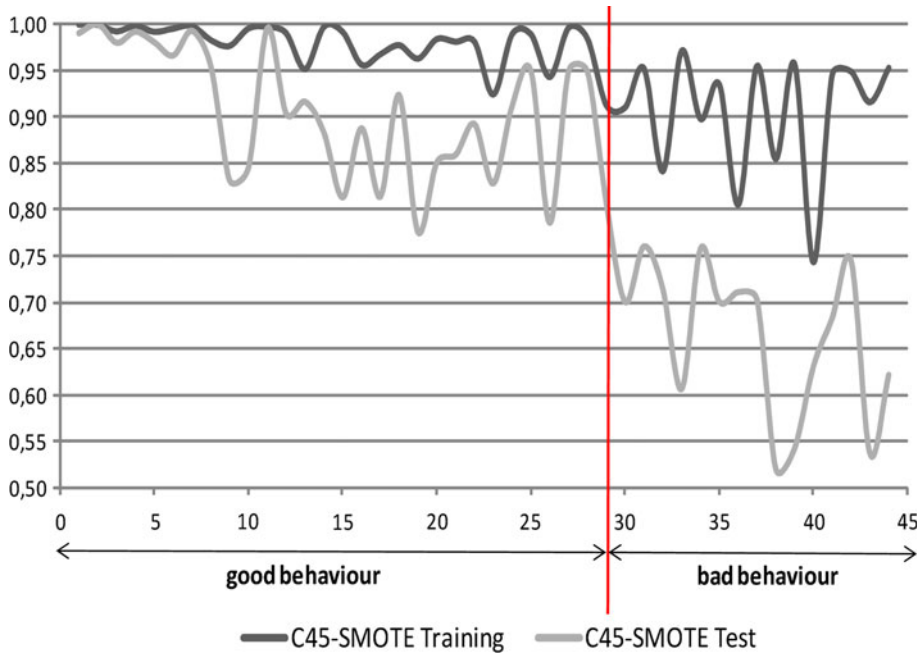

Fig. 15 C4.5 with SMOTE AUC in Training/Test sorted by N4

C4.5 and PART considering any preprocessing technique.

- SMOTE and SMOTE-ENN behave equal for C4.5 and PART. The differences with respect to the global training and test AUC are similar, and the support is equal in every case.
- EUSCHC behaves equally for C4.5 and PART when considering the N4 and L3 data complexity measures. However, for the F1 measure, EUSCHC is slightly better for PART, as the support of R1+ for this learning method is a 3% higher.

5.3 Combination of the single rules

The objective of this section is to analyze the effect of combining the rules of good and behavior independently. By means of merging the individual rules we can arrive at a

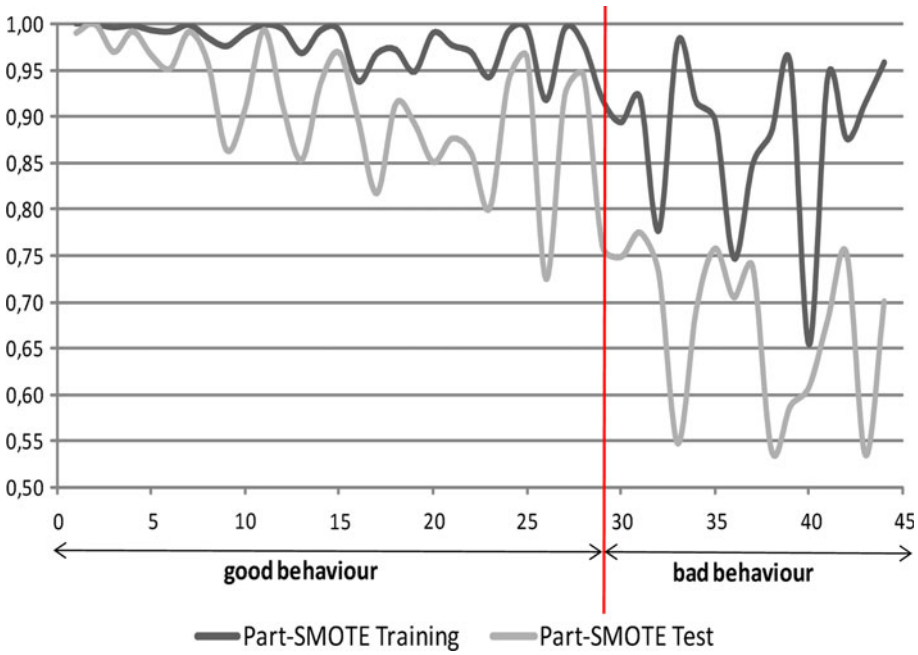

Fig. 16 PART with SMOTE AUC in Training/Test sorted by N4

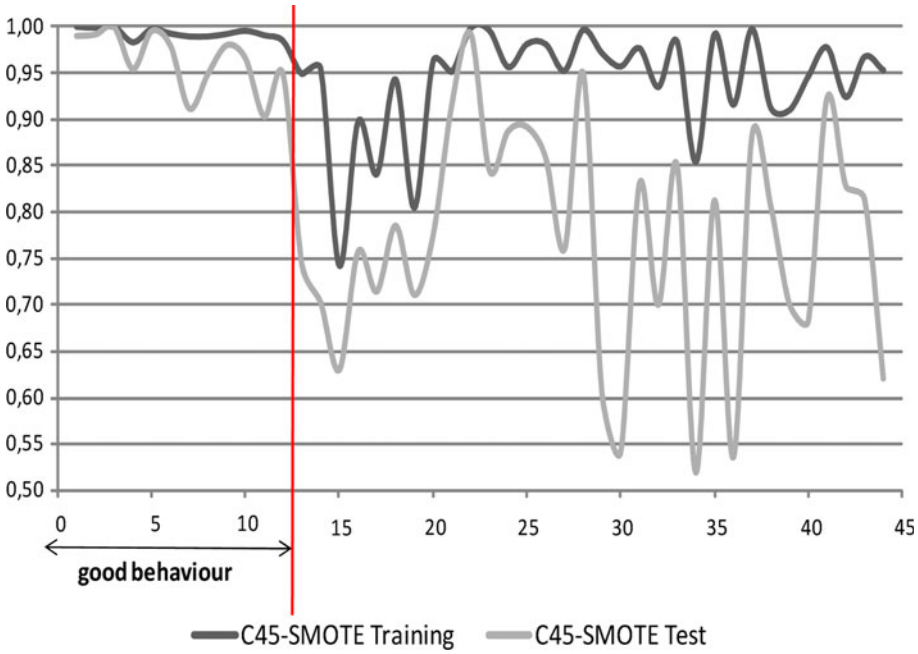

Fig. 17 C4.5 with SMOTE AUC in Training/Test sorted by L3

more general description, with a wider support, of the preprocessing methods' effect.

First, we have considered the disjunctive combination of all the positive rules to obtain a single rule (Positive Rule Disjunction -PRD-), that is, we use the *or* operator to combine the individual positive rules. The same procedure is done with all the negative ones so we obtain another rule (Negative Rule Disjunction -NRD-). The new disjunctive rules will be activated if any of the component rules' antecedents are verified.

In the case of PART with EUSCHC preprocessing, overlapping between the data sets covered by PRD and NRD appears. Following the same methodology presented by Luengo and Herrera (2010) the PRD rule is kept as representative of the good behavior, and the set difference between the NRD and PRD rules is

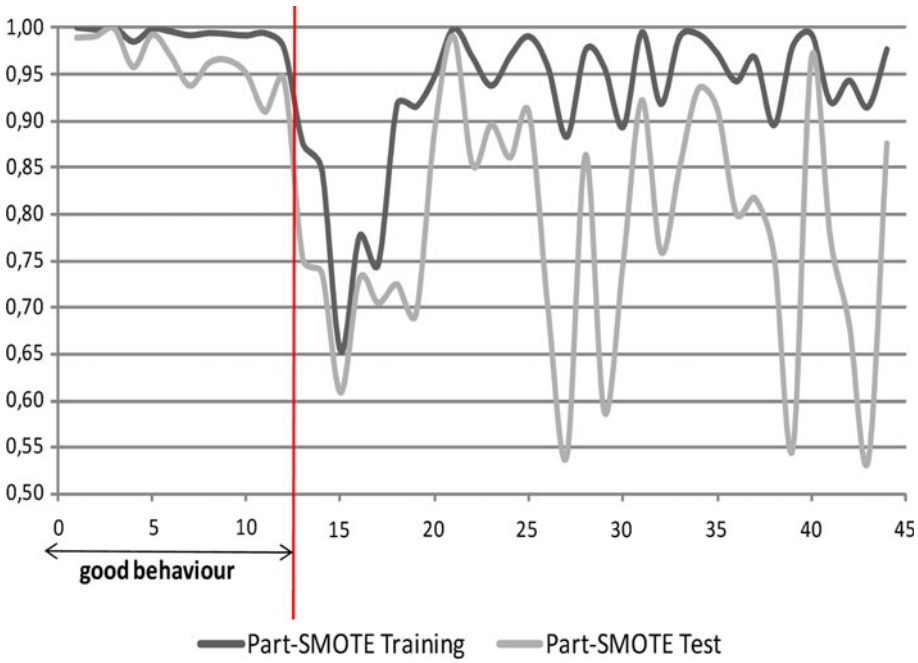

Fig. 18 PART with SMOTE AUC in Training/Test sorted by L3

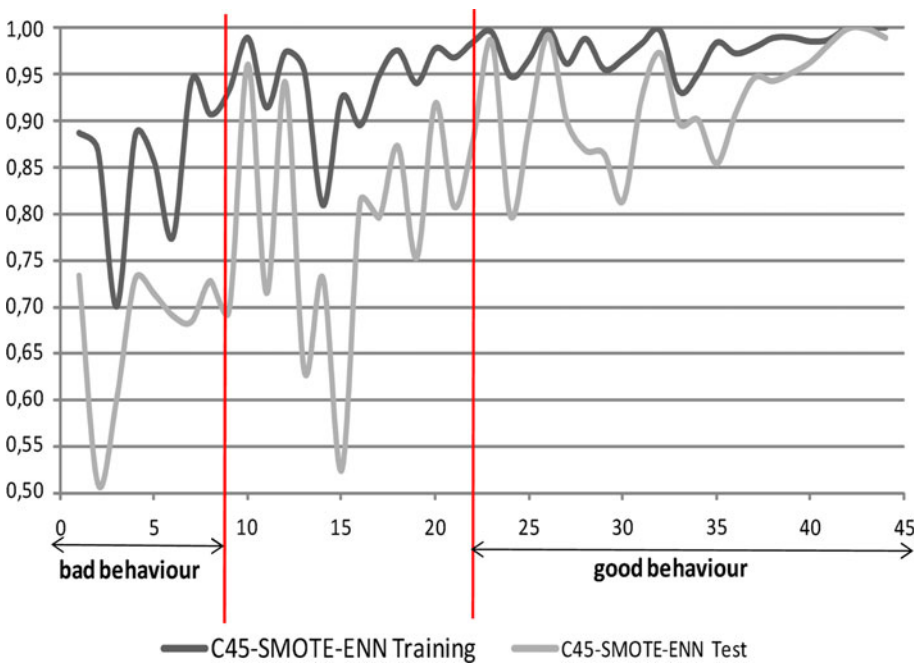

Fig. 19 C4.5 with SMOTE-ENN AUC in Training/Test sorted by F1

considered as representative of the bad behavior. This difference will remove the data sets for which C4.5 and PART presents good behavior from the NRD rule, naming this new rule as NRD$\wedge\neg$PRD.

In Table 12 we summarize the new rules obtained for C4.5. In Table 13 the equivalent rules for PART are depicted. In these two tables, we have added the following suffixes to the rule identifier in order to distinguish them:

- In the case of SMOTE preprocessing, we add the "-S" suffix.
- In the case of SMOTE-ENN preprocessing, we add the "-S-ENN" suffix.

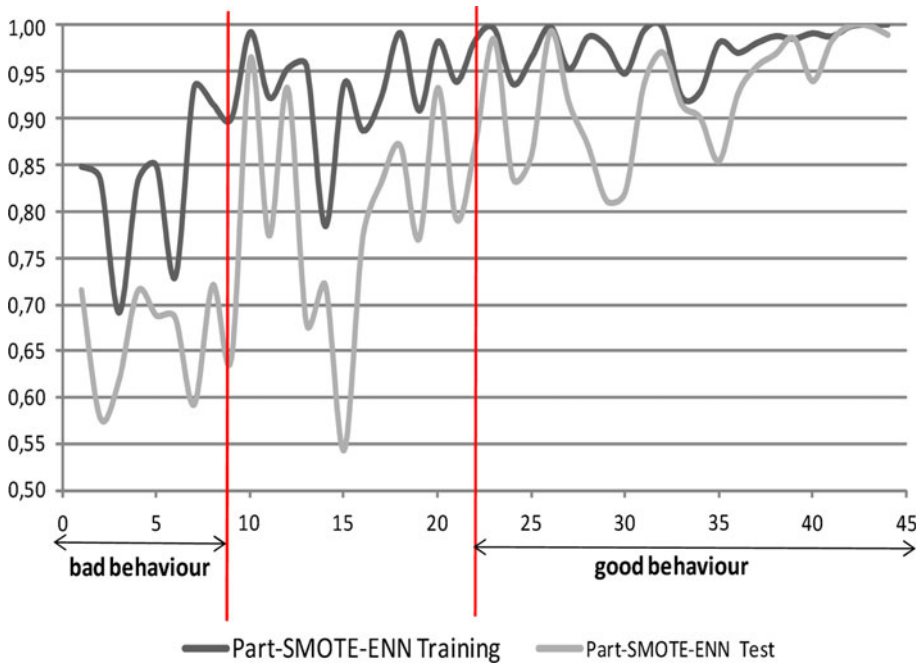

Fig. 20 PART with SMOTE-ENN AUC in Training/Test sorted by F1

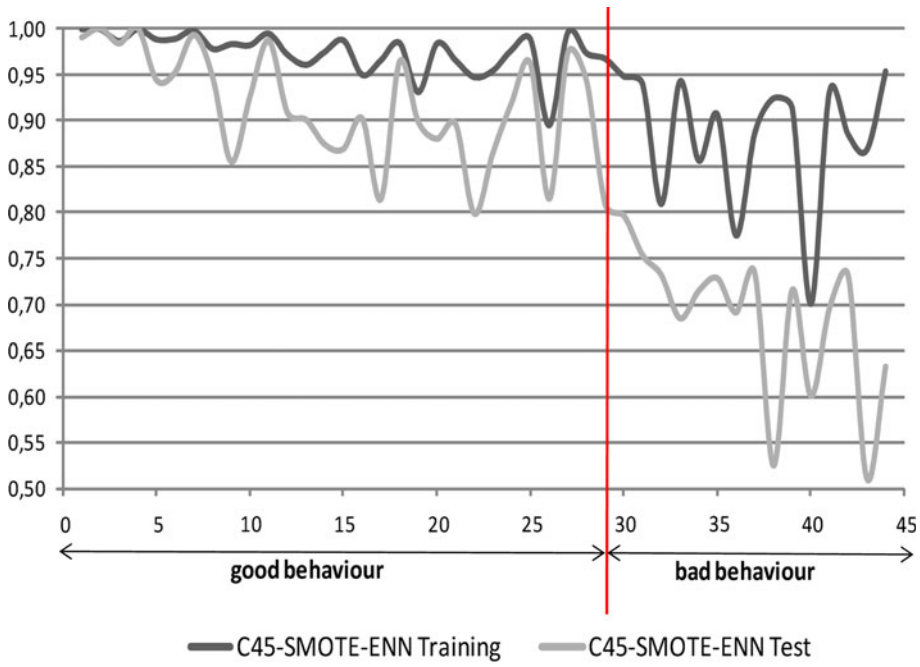

Fig. 21 C4.5 with SMOTE-ENN AUC in Training/Test sorted by N4

- In the case of EUSCHC preprocessing, we add the "-EUS" suffix.

From the collective rules for both learning methods, it is observed that the support has been increased from the single rules for PRD, while NRD (and NRD∧¬PRD for EUSCHC) obtains similar support. On the other hand, the training and test AUC differences are similar to the single rules from Tables 7 and 8 in both with and without preprocessing situations.

With the PRD and NRD (NRD∧¬PRD-EUS for PART) there is no uncovered data sets by the rules for C4.5 and

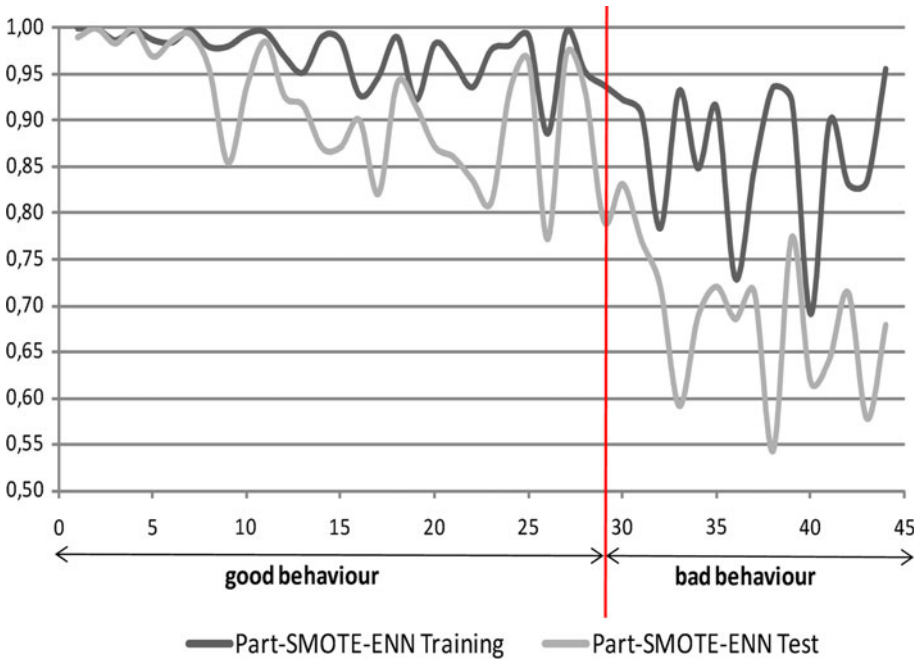

Fig. 22 PART with SMOTE-ENN AUC in Training/Test sorted by N4

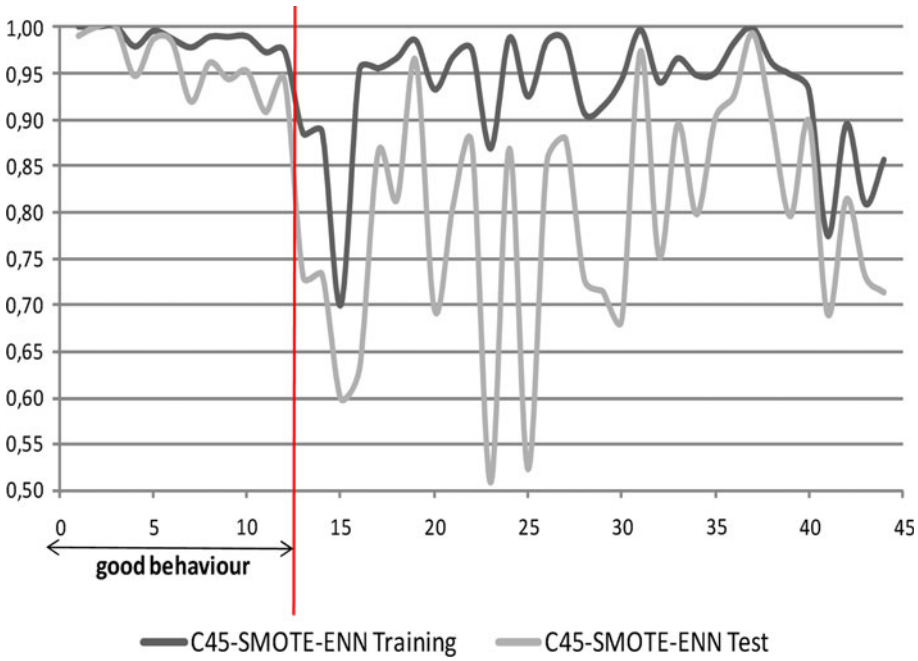

Fig. 23 C4.5 with SMOTE-ENN AUC in Training/Test sorted by L3

PART in combination with SMOTE, SMOTE-ENN, and EUSCHC preprocessing methods. Therefore, we can consider a two block representation of the data sets.

- The first block (the left-side one) will represent the data sets covered by the correspondent PRD rule. They are the data sets recognized as being those in which C4.5 and PART have good AUC when preprocessing.
- The second (the right-side one) will plot the data sets for the correspondent NRD (NRD$\wedge\neg$PRD-EUS for PART) rule, which are bad data sets for C4.5 and PART after preprocessing.

In Figs. 7, 9 and 11 we have depicted the two block representation for C4.5 considering the three cases of preprocessing. In Figs. 8, 10 and 12 we have depicted the

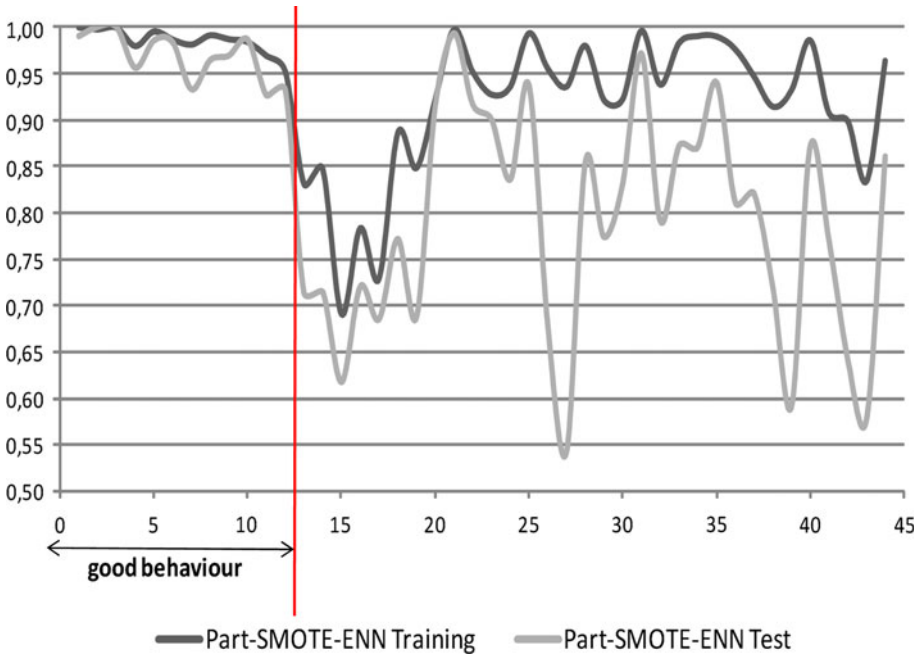

Fig. 24 PART with SMOTE-ENN AUC in Training/Test sorted by L3

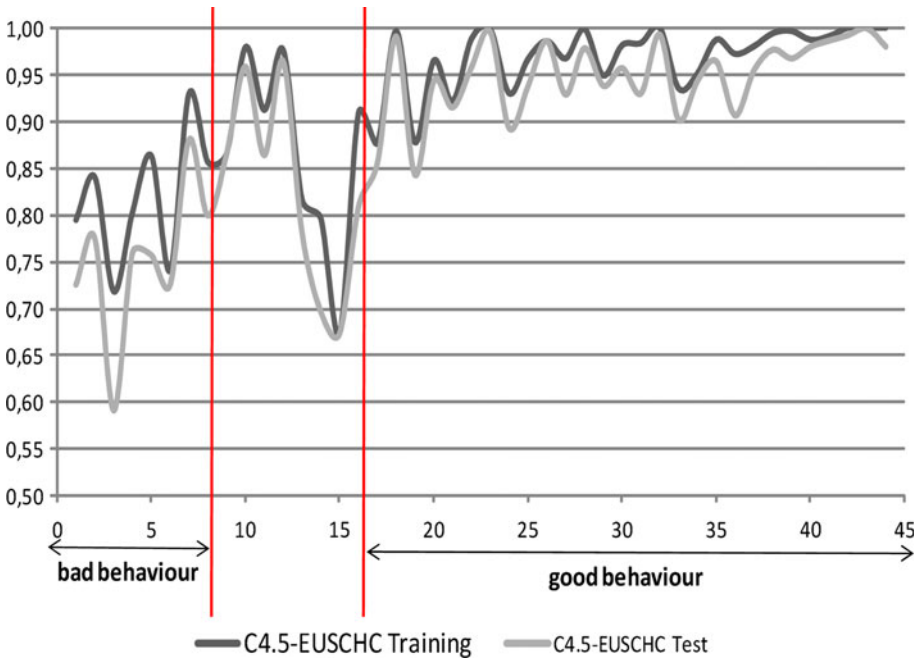

Fig. 25 C4.5 with EUSCHC AUC in Training/Test sorted by F1

same representations for PART. The data sets have the same order in the X axis in all the figures to facilitate the comparisons.

Table 14 represents the data sets following the order of the latter figures indicating those data sets which are covered by the PRD and NRD (NRD$\wedge\neg$PRD-EUS for PART) rules as indicate by the vertical lines in the two blocks representation.

We can observe that the 100% of the analyzed data sets are covered by the two considered rules for each preprocessing method. Since SMOTE and SMOTE-ENN obtained the same intervals in the previous subsection, the

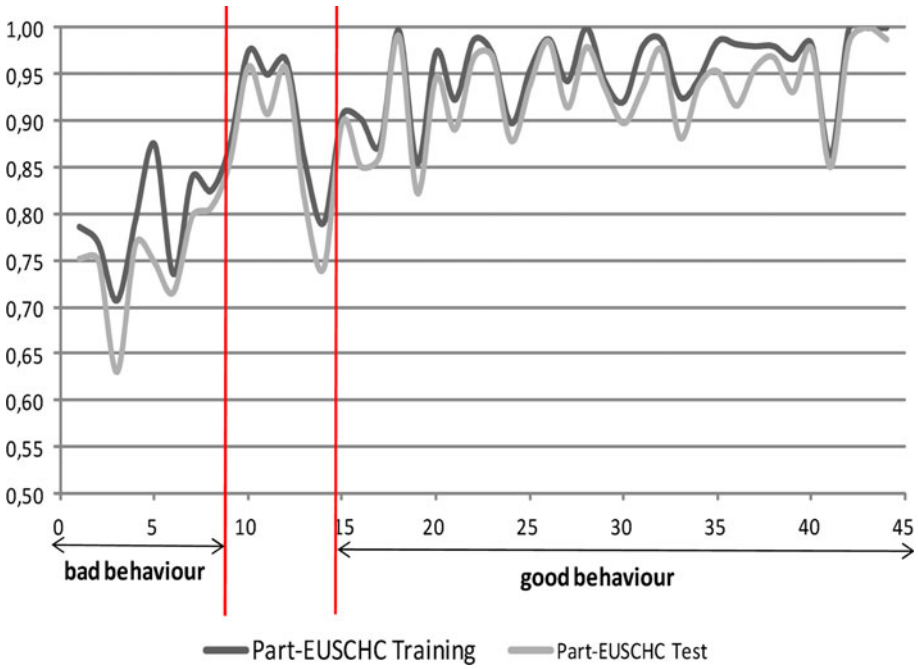

Fig. 26 PART with EUSCHC AUC in Training/Test sorted by F1

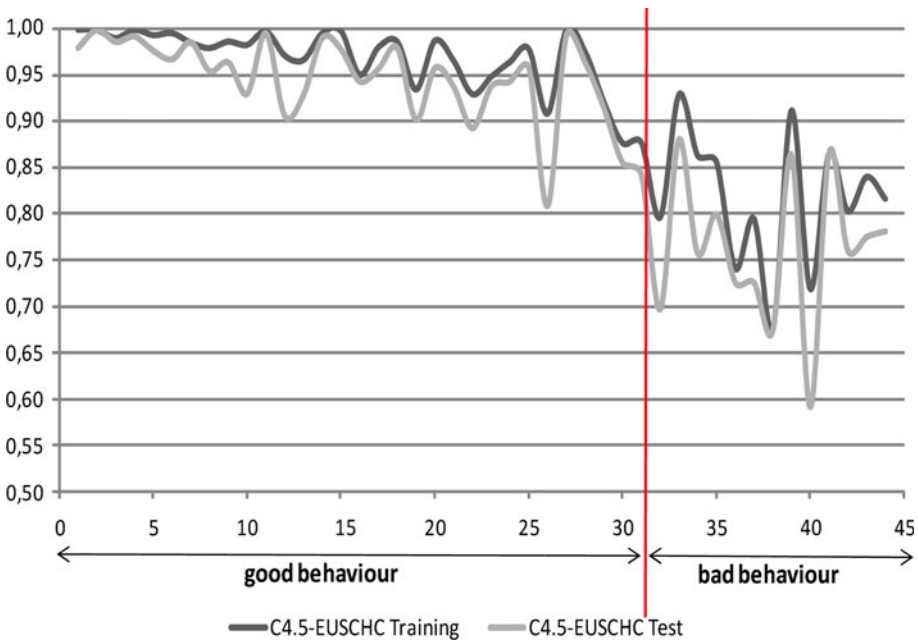

Fig. 27 C4.5 with EUSCHC AUC in Training/Test sorted by N4

support of the PRD and NRD rules are the same, and they cover the same data sets.

The EUSCHC approach obtains a wider support for the PRD rule with respect to the SMOTE and SMOTE-ENN approaches. This is due to the wider support of the individual intervals which conform the PRD rule. This difference indicates that the undersampling approach is more beneficial for C4.5 and PART, since more data sets are characterized as good for these two learning methods.

From these results we can point out that the data complexity measures are useful to evaluate the behavior of the undersampling and oversampling approaches. Differences in their results have been characterized, finding that

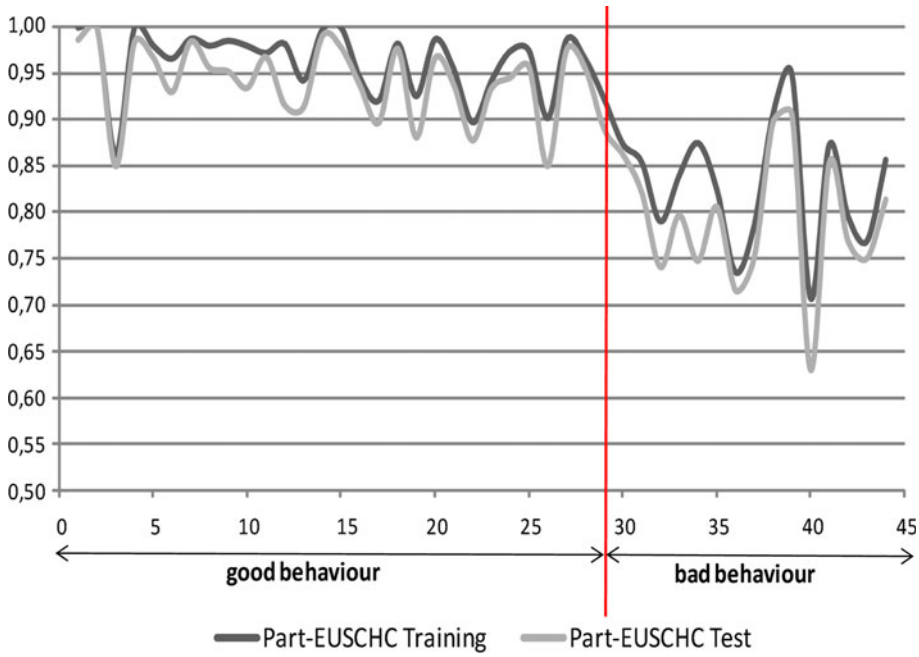

Fig. 28 PART with EUSCHC AUC in Training/Test sorted by N4

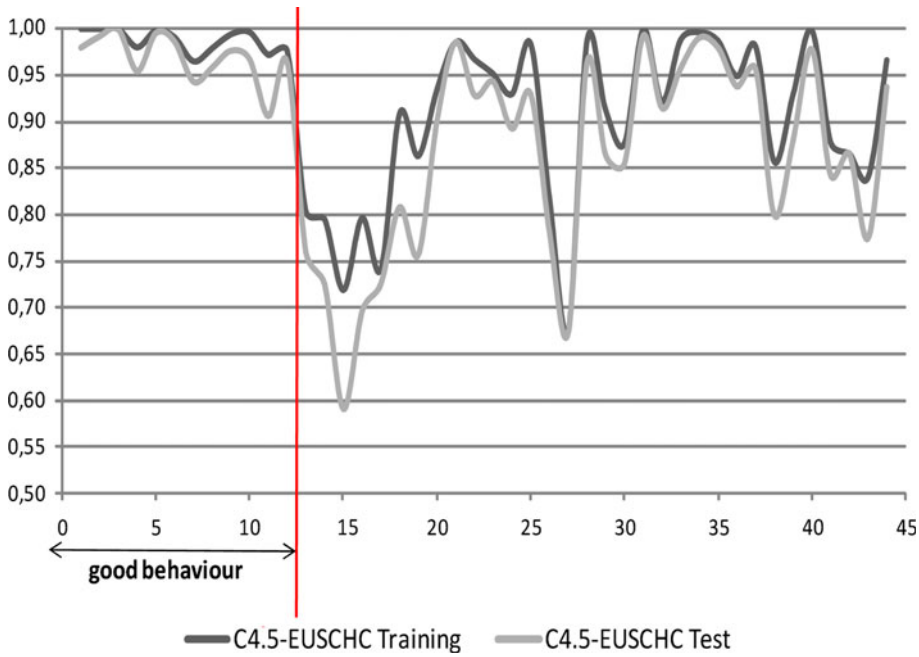

Fig. 29 C4.5 with EUSCHC AUC in Training/Test sorted by L3

EUSCHC is more robust than SMOTE-based approaches due to its wider region of good behavior.

There is a bunch of data sets for which both undersampling and oversampling techniques do not work well (indicated in Table 14), as the data set covered by all NRD and the NRD$\wedge\neg$PRD-EUS rule. These data sets are therefore opened to improvements by means of these or other techniques, but already identified by the rules.

6 Concluding remarks

In this work we have analyzed the preprocessing effect in the framework of imbalanced data sets by means of data complexity measures. We have considered two oversampling methods: SMOTE and SMOTE-ENN, and an evolutionary undersampling approach: EUSCHC.

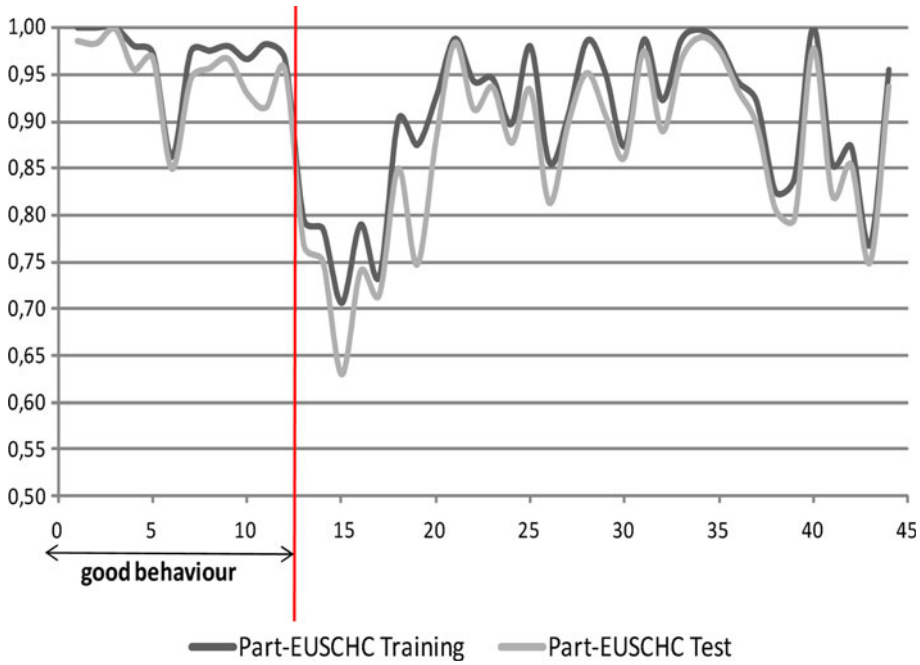

Fig. 30 PART with EUSCHC AUC in Training/Test sorted by L3

We have observed that the IR considered as a measure of data complexity is not enough to predict when C4.5 and PART perform good or bad. As an alternative approach, we have computed the data complexity measures over the imbalanced data sets in order to obtain intervals of such metrics in which C4.5 and PART performance is significantly good and bad when using the three preprocessing methods. From these intervals we have built descriptive rules, which have a wide support and a significative difference with respect to the global methods' performance.

We have obtained two final rules from the initial ones, which are simple and precise to describe both good and bad performance of C4.5 and PART. These two rules are capable of identifying all good and bad data sets for SMOTE, SMOTE-ENN, and EUSCHC. An interesting consequence of the characterization obtained by the rules is that the evolutionary undersampling approach is capable of preprocessing successfully more data sets for C4.5 and PART.

As a final note, it is interesting to indicate that the Fisher's Discriminant Ratio (F1) was also found interesting by the studies of Mollineda et al. (2005); Kim and Oommen (2009) considering prototype selection, and it is informative for our analysis in the imbalance framework as well.

Acknowledgments This work has been supported by the Spanish Ministry of Education and Science under Project TIN2008-06681-C06-(01 and 02). J. Luengo holds a FPU scholarship from Spanish Ministry of Education.

Appendix 1: Figures with the intervals of PART and C4.5

In this appendix, the figures sorted by the F1, N4 and L3 data complexity measures are depicted. We have used a two-column representation for the figures, so in each row we present the results for C4.5 and PART for the same case of type of preprocessing and data complexity measure used.

- Figures from 13, 14, 15, 16, 17, 18 represents the figures for the case of SMOTE preprocessing.
- Figures from 19, 20, 21, 22, 23, 24 represents the figures for the case of SMOTE-ENN preprocessing.
- Figures from 25, 26, 27, 28, 29, 30 represents the figures for the case of EUSCHC preprocessing.

Appendix 2: Tables of results

In this appendix we present the average AUC results for C4.5 and PART in Tables 15 and 16 respectively.

Table 15 Average AUC results for C4.5

Data sets	SMOTE Training	SMOTE Test	SMOTE-ENN Training	SMOTE-ENN Test	EUSCHC Training	EUSCHC Test
Ecoli0vs1	0.9927	0.9796	0.9870	0.9832	0.9909	0.9864
Haberman	0.7426	0.6309	0.6999	0.6003	0.7190	0.5914
Iris0	1.0000	0.9900	1.0000	0.9900	1.0000	0.9800
Pima	0.8411	0.7145	0.8089	0.7312	0.7966	0.6966
Vehicle2	0.9895	0.9492	0.9890	0.9611	0.9793	0.9586
Wisconsin	0.9832	0.9545	0.9784	0.9467	0.9802	0.9546
Yeast2	0.8049	0.7109	0.7744	0.6904	0.7411	0.7257
Glass0	0.9433	0.7856	0.8950	0.8143	0.9089	0.8085
Glass1	0.8978	0.7577	0.8563	0.7141	0.8633	0.7571
Vehicle1	0.9551	0.7030	0.8866	0.7335	0.7952	0.7258
Vehicle3	0.9493	0.7444	0.8844	0.7304	0.8035	0.7601
Ecoli1	0.9631	0.7755	0.9313	0.8979	0.9353	0.9019
Glass0123vs456	0.9908	0.9032	0.9721	0.9078	0.9720	0.9063
New-Thyroid1	0.9922	0.9802	0.9888	0.9433	0.9942	0.9767
New-Thyroid2	0.9957	0.9659	0.9895	0.9520	0.9965	0.9674
Page-Blocks0	0.9846	0.9485	0.9737	0.9421	0.9763	0.9644
Segment0	0.9985	0.9927	0.9985	0.9927	0.9859	0.9861
Vehicle0	0.9897	0.9118	0.9775	0.9192	0.9651	0.9433
Ecoli2	0.9517	0.9162	0.9610	0.9002	0.9673	0.9287
Yeast3	0.9565	0.8876	0.9500	0.9016	0.9513	0.9434
Ecoli3	0.9815	0.8921	0.9474	0.7980	0.9301	0.8928
Glass6	0.9959	0.8450	0.9825	0.9257	0.9836	0.9299
Abalone9-18	0.9531	0.6215	0.9539	0.6322	0.8167	0.7812
Abalone19	0.8544	0.5202	0.9245	0.5246	0.6751	0.6736
Ecoli4	0.9769	0.8310	0.9839	0.8544	0.9873	0.9643
Glass2	0.9571	0.5424	0.9139	0.7148	0.9124	0.8640
Yeast4	0.9101	0.7004	0.9484	0.7960	0.8770	0.8551
Vowel0	0.9967	0.9494	0.9965	0.9733	1.0000	0.9929
Yeast2vs8	0.9125	0.8066	0.9677	0.8078	0.9217	0.9149
Glass4	0.9844	0.8508	0.9844	0.8794	0.9883	0.9580
Glass5	0.9976	0.8829	0.9753	0.8732	0.9965	0.9907
Yeast5	0.9777	0.9233	0.9851	0.9635	0.9870	0.9798
Yeast6	0.9242	0.8280	0.9549	0.8647	0.9491	0.9380
Ecoli0137vs26	0.9678	0.8136	0.9660	0.8136	0.9813	0.9572
Shuttle0vs4	0.9999	0.9997	0.9999	0.9997	1.0000	0.9995
YeastB1vs7	0.9351	0.7003	0.9066	0.7278	0.8567	0.7996
Shuttle2vs4	0.9990	0.9917	1.0000	1.0000	1.0000	0.9923
Glass016vs2	0.9716	0.6062	0.9430	0.6840	0.9296	0.8806
Glass016vs5	0.9921	0.8129	0.9879	0.8686	0.9986	0.9784
Page-Blocks13vs4	0.9975	0.9955	0.9952	0.9865	0.9984	0.9957
Yeast05679vs4	0.9526	0.7602	0.9401	0.7527	0.8778	0.8429
Yeast1289vs7	0.9465	0.6832	0.9323	0.6955	0.8659	0.8659
Yeast1458vs7	0.9158	0.5367	0.8685	0.5102	0.8405	0.7750
Yeast2vs4	0.9814	0.8588	0.9659	0.8953	0.9664	0.9377
Global	0.9546	0.8217	0.9438	0.8362	0.9241	0.8914

Table 16 Average AUC results for PART

Data sets	SMOTE Training	SMOTE Test	SMOTE-ENN Training	SMOTE-ENN Test	EUSCHC Training	EUSCHC Test
Ecoli0vs1	0.9958	0.9694	0.9870	0.9832	0.8625	0.8500
Haberman	0.6540	0.6086	0.6909	0.6183	0.7067	0.6305
Iris0	1.0000	0.9900	1.0000	0.9900	1.0000	0.9867
Pima	0.7769	0.7312	0.7836	0.7209	0.7900	0.7409
Vehicle2	0.9942	0.9628	0.9917	0.9642	0.9752	0.9574
Wisconsin	0.9848	0.9584	0.9800	0.9559	0.9802	0.9561
Yeast2	0.7468	0.7049	0.7288	0.6858	0.7350	0.7156
Glass0	0.9176	0.7250	0.8861	0.7720	0.9019	0.8503
Glass1	0.9151	0.6927	0.8485	0.6880	0.8750	0.7477
Vehicle1	0.8484	0.7377	0.8475	0.7153	0.7861	0.7518
Vehicle3	0.8757	0.7519	0.8314	0.7144	0.7949	0.7683
Ecoli1	0.9480	0.8923	0.9226	0.9151	0.9256	0.8810
Glass0123vs456	0.9939	0.9104	0.9695	0.9262	0.9825	0.9157
New-Thyroid1	0.9930	0.9659	0.9874	0.9690	0.9802	0.9674
New-Thyroid2	0.9915	0.9516	0.9845	0.9861	0.9663	0.9302
Page-Blocks0	0.9774	0.9439	0.9529	0.9322	0.9657	0.9556
Segment0	0.9987	0.9911	0.9978	0.9932	0.9880	0.9848
Vehicle0	0.9916	0.9382	0.9815	0.9328	0.9743	0.9456
Ecoli2	0.9681	0.8533	0.9521	0.9164	0.9427	0.9137
Yeast3	0.9377	0.8966	0.9277	0.9005	0.9456	0.9373
Ecoli3	0.9693	0.8611	0.9361	0.8359	0.8974	0.8779
Glass6	0.9905	0.9090	0.9939	0.9369	0.9802	0.9344
Abalone9-18	0.9581	0.7006	0.9559	0.6794	0.8567	0.8139
Abalone19	0.8831	0.5401	0.9362	0.5434	0.9066	0.8979
Ecoli4	0.9757	0.8639	0.9804	0.8544	0.9859	0.9524
Glass2	0.9571	0.5878	0.9218	0.7742	0.9497	0.9066
Yeast4	0.8936	0.7486	0.9228	0.8316	0.8738	0.8625
Vowel0	0.9950	0.9228	0.9967	0.9711	0.9868	0.9757
Yeast2vs8	0.9182	0.7599	0.9384	0.7915	0.9227	0.8901
Glass4	0.9901	0.8508	0.9832	0.8718	0.9872	0.9670
Glass5	0.9927	0.9354	0.9909	0.8707	0.9977	0.9907
Yeast5	0.9721	0.9132	0.9905	0.9403	0.9826	0.9771
Yeast6	0.9424	0.8008	0.9767	0.8115	0.9422	0.9340
Ecoli0137vs26	0.9678	0.8172	0.9474	0.8209	0.9208	0.8969
Shuttle0vs4	0.9999	0.9997	0.9999	0.9997	1.0000	0.9997
YeastB1vs7	0.8954	0.7576	0.9147	0.7207	0.8246	0.8061
Shuttle2vs4	0.9980	0.9917	0.9980	1.0000	1.0000	0.9840
Glass016vs2	0.9800	0.5479	0.9323	0.5921	0.8397	0.7969
Glass016vs5	0.9929	0.9686	0.9864	0.8714	1.0000	0.9784
Page-Blocks13vs4	0.9986	0.9932	0.9958	0.9854	0.9725	0.9681
Yeast05679vs4	0.9204	0.7748	0.9076	0.7704	0.8546	0.8221
Yeast1289vs7	0.9433	0.6815	0.8992	0.6427	0.8735	0.8543
Yeast1458vs7	0.9151	0.5351	0.8343	0.5783	0.7688	0.7503
Yeast2vs4	0.9765	0.8762	0.9642	0.8607	0.9548	0.9377
Global	0.9440	0.8298	0.9353	0.8372	0.9172	0.8900

References

Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) KEEL: A software tool to assess evolutionary algorithms to data mining problems. Soft Comput 13(3):307–318

Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html

Barandela R, Sánchez JS, García V, Rangel E (2003) Strategies for learning in class imbalance problems. Pattern Recognit 36(3):849–851

Basu M, Ho TK (2006) Data complexity in pattern recognition (advanced information and knowledge processing). Springer-Verlag New York, Inc., Secaucus

Batista GEAPA, Prati RC, Monard MC (2004) A study of the behaviour of several methods for balancing machine learning training data. SIGKDD Explor 6(1):20–29

Baumgartner R, Somorjai RL (2006) Data complexity assessment in undersampled classification of high-dimensional biomedical data. Pattern Recognit Lett 12:1383–1389

Bernadó-Mansilla E, Ho TK (2005) Domain of competence of XCS classifier system in complexity measurement space. IEEE Trans Evol Comput 9(1):82–104

Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159

Brazdil P, Giraud-Carrier C, Soares C, Vilalta R (2009) Metalearning: applications to data mining. Cognitive Technologies, Springer. http://10.255.0.115/pub/2009/BGSV09

Celebi M, Kingravi H, Uddin B, Iyatomi H, Aslandogan Y, Stoecker W, Moss R (2007) A methodological approach to the classification of dermoscopy images. Comput Med Imaging Graphics 31(6):362–373

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Chawla NV, Japkowicz N, Kolcz A (2004) Editorial: special issue on learning from imbalanced data sets. SIGKDD Explor 6(1):1–6

Diamantini C, Potena D (2009) Bayes vector quantizer for class-imbalance problem. IEEE Trans Knowl Data Eng 21(5):638–651

Domingos P (1999) Metacost: a general method for making classifiers cost sensitive. In: Advances in neural networks, Int J Pattern Recognit Artif Intell, pp 155–164

Dong M, Kothari R (2003) Feature subset selection using a new definition of classificabilty. Pattern Recognit Lett 24:1215–1225

Drown DJ, Khoshgoftaar TM, Seliya N (2009) Evolutionary sampling and software quality modeling of high-assurance systems. IEEE Trans Syst Man Cybern A 39(5):1097–1107

Eshelman LJ (1991) Foundations of genetic algorithms, chap The CHC adaptive search algorithm: how to safe search when engaging in nontraditional genetic recombination, pp 265–283

Estabrooks A, Jo T, Japkowicz N (2004) A multiple resampling method for learning from imbalanced data sets. Comput Intell 20(1):18–36

Fernández A, García S, del Jesus MJ, Herrera F (2008) A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. Fuzzy Sets Syst 159(18):2378–2398

Frank E, Witten IH (1998) Generating accurate rule sets without global optimization. In: ICML '98: Proceedings of the fifteenth international conference on machine learning, Morgan Kaufmann Publishers Inc., San Francisco, pp 144–151

García S, Herrera F (2009a) Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy. Evol Comput 17(3):275–306

García S, Fernández A, Herrera F (2009b) Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. Appl Soft Comput 9(4):1304–1314

García S, Cano JR, Bernadó-Mansilla E, Herrera F (2009c) Diagnose of effective evolutionary prototype selection using an overlapping measure. Int J Pattern Recognit Artif Intell 23(8):2378–2398

García V, Mollineda R, Sánchez JS (2008) On the k–NN performance in a challenging scenario of imbalance and overlapping. Pattern Anal Appl 11(3–4):269–280

He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284

Ho TK, Basu M (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 24(3):289–300

Hoekstra A, Duin RP (1996) On the nonlinearity of pattern classifiers. In: ICPR '96: Proceedings of the international conference on pattern recognition (ICPR '96) Volume IV-Volume 7472, IEEE Computer Society, Washington, DC, pp 271–275

Huang J, Ling CX (2005) Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng 17(3):299–310

Kalousis A (2002) Algorithm selection via meta-learning. PhD thesis, Université de Geneve

Kilic K, Uncu O, Türksen IB (2007) Comparison of different strategies of utilizing fuzzy clustering in structure identification. Inform Sci 177(23):5153–5162

Kim SW, Oommen BJ (2009) On using prototype reduction schemes to enhance the computation of volume-based inter-class overlap measures. Pattern Recognit 42(11):2695–2704

Li Y, Member S, Dong M, Kothari R, Member S (2005) Classifiability-based omnivariate decision trees. IEEE Trans Neural Netw 16(6):1547–1560

Lu WZ, Wang D (2008) Ground-level ozone prediction by support vector machine approach with a cost-sensitive classification scheme. Sci Total Environ 395(2–3):109–116

Luengo J, Herrera F (2010) Domains of competence of fuzzy rule based classification systems with data complexity measures: A case of study using a fuzzy hybrid genetic based machine learning method. Fuzzy Sets Syst 161(1):3–19

Mazurowski M, Habas P, Zurada J, Lo J, Baker J, Tourassi G (2008) Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance. Neural Netw 21(2–3):427–436

Mollineda RA, Sánchez JS, Sotoca JM (2005) Data characterization for effective prototype selection. In: First edition of the Iberian conference on pattern recognition and image analysis (IbPRIA 2005), Lecture Notes in Computer Science 3523, pp 27–34

Orriols-Puig A, Bernadó-Mansilla E (2008) Evolutionary rule-based systems for imbalanced data sets. Soft Comput 13(3):213–225

Peng X, King I (2008) Robust BMPM training based on second-order cone programming and its application in medical diagnosis. Neural Netw 21(2–3):450–457

Pfahringer B, Bensusan H, Giraud-Carrier CG (2000) Meta-learning by landmarking various learning algorithms. In: ICML '00: Proceedings of the seventeenth international conference on machine learning, Morgan Kaufmann Publishers Inc., San Francisco, pp 743–750

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

Sánchez J, Mollineda R, Sotoca J (2007) An analysis of how training data complexity affects the nearest neighbor classifiers. Pattern Anal Appl 10(3):189–201

Singh S (2003) Multiresolution estimates of classification complexity. IEEE Trans Pattern Anal Mach Intell 25(12):1534–1539

Su CT, Hsiao YH (2007) An evaluation of the robustness of MTS for imbalanced data. IEEE Trans Knowl Data Eng 19(10):1321–1332

Sun Y, Kamel MS, Wong AKC, Wang Y (2007) Cost–sensitive boosting for classification of imbalanced data. Pattern Recognit 40:3358–3378

Sun Y, Wong AKC, Kamel MS (2009) Classification of imbalanced data: A review. Int J Pattern Recognit Artif Intell 23(4):687–719

Tang Y, Zhang YQ, Chawla N (2009) SVMs modeling for highly imbalanced classification. IEEE Trans Syst Man Cybern B Cybern 39(1):281–288

Williams D, Myers V, Silvious M (2009) Mine classification with imbalanced data. IEEE Geosci Remote Sens Lett 6(3):528–532

Yang Q, Wu X (2006) 10 challenging problems in data mining research. Int J Inform Tech Decis Mak 5(4):597–604

Zhou ZH, Liu XY (2006) Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans Knowl Data Eng 18(1):63–77

Bibliography

[ACW06] Au W.-H., Chan K. C. C., y Wong A. K. C. (2006) A fuzzy approach to partitioning continuous attributes for classification. *IEEE Transactions on Knowledge and Data Engineering* 18(5): 715–719.

[AMS97] Atkeson C. G., Moore A. W., y Schaal S. (1997) Locally weighted learning. *Artificial Intelligence Review* 11: 11–73.

[BGCSV09] Brazdil P., Giraud-Carrier C., Soares C., y Vilalta R. (January 2009) *Metalearning: Applications to Data Mining*. Cognitive Technologies. Springer.

[BH98] Berthold M. R. y Huber K.-P. (April 1998) Missing values and learning of fuzzy rules. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6: 171–178.

[BK01a] Bensusan H. y Kalousis A. (2001) Estimating the predictive accuracy of a classifier. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pp. 25–36. Springer-Verlag, London, UK.

[BK01b] Bezdek J. C. y Kuncheva L. (2001) Nearest prototype classifier designs: An experimental study. *International Journal of Intelligent Systems* 16(12): 1445–1473.

[BL88] Broomhead D. y Lowe D. (1988) Multivariable functional interpolation and adaptive networks. *Complex Systems* 2: 321–355.

[BM99] Barnard J. y Meng X. (1999) Applications of multiple imputation in medical studies: From AIDS to NHANES. *Statistical methods in medical research* 8(1): 17–36.

[BM03] Batista G. y Monard M. (2003) An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence* 17(5): 519–533.

[BMH05] Bernadó-Mansilla E. y Ho T. K. (2005) Domain of competence of xcs classifier system in complexity measurement space. *IEEE Transactions on Evolutionary Computation* 9(1): 82–104.

[BPM04] Batista G. E. A. P. A., Prati R. C., y Monard M. C. (2004) A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter* 6(1): 20–29.

[BS04] Baskiotis N. y Sebag M. (2004) C4.5 competence map: a phase transition-inspired approach. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. ACM, New York, NY, USA.

[Buh03] Buhmann M. D. (2003) *Radial Basis Functions: Theory and Implementations.* Brooks Cole.

[CBHK02] Chawla N. V., Bowyer K. W., Hall L. O., y Kegelmeyer W. P. (2002) Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16: 321–357.

[CCHJ08] Chawla N. V., Cieslak D. A., Hall L. O., y Joshi A. (2008) Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery* 17(2): 225–252.

[CdJH99] Cordón O., del Jesus M., y Herrera F. (1999) A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning* 20(1): 21–45.

[CHV00] Cordón O., Herrera F., y Villar P. (2000) Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing. *International Journal on Approximate Reasoning* 25(3): 187–215.

[CJK04] Chawla N. V., Japkowicz N., y Kolcz A. (2004) Special issue on learning from imbalanced datasets. *SIGKDD Explorations Newsletter* 6(1).

[CN89] Clark P. y Niblett T. (1989) The cn2 induction algorithm. *Machine Learning Journal* 3(4): 261–283.

[Coh95] Cohen W. W. (1995) Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 115–123. Morgan Kaufmann.

[CS99] Cohen W. y Singer Y. (1999) A simple and fast and and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp. 335–342.

[CV95] Cortes C. y Vapnik V. (1995) Support vector networks. *Machine Learning* 20: 273–297.

[CW03] Chen Y. y Wang J. Z. (2003) Support vector learning for fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems* 11(6): 716–728.

[Dem06] Demšar J. (2006) Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7: 1–30.

[DP97] Domingos P. y Pazzani M. (1997) On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning* 29: 103–137.

[EFW01] Ennett C. M., Frize M., y Walker C. R. (2001) Influence of missing values on artificial neural network performance. *Studies in health technology and informatics* 84: 449–453.

[EJJ04] Estabrooks A., Jo T., y Japkowicz N. (2004) A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence* 20(1): 18–36.

[FCL05] Fan R.-E., Chen P.-H., y Lin C.-J. (2005) Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research* 6: 1889–1918.

[FHOR05] Ferri C., Hernández-Orallo J., y Ramírez M. J. (2005) *Introducción a la Minería de Datos*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[FKD08] Farhangfar A., Kurgan L., y Dy J. (2008) Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition* 41(12): 3692–3705.

[FKP07] Farhangfar A., Kurgan L. A., y Pedrycz W. (2007) A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 37(5): 692–709.

[FW98] Frank E. y Witten I. (1998) Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 144–151.

[GB05] Gabriel T. R. y Berthold M. R. (2005) Missing values in fuzzy rule induction. In Anderson G. y Tunstel E. (Eds.) *2005 IEEE Conference on Systems, Man and Cybernetics*. IEEE Press.

[GH08] García S. y Herrera F. (2008) An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research* 9: 2677–2694.

[GH09] García S. y Herrera F. (2009) Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation* 17(3): 275–306.

[GLSGFV09] García-Laencina P., Sancho-Gómez J., y Figueiras-Vidal A. (2009) Pattern classification with missing data: a review. *Neural Computation & Applications* 9(1): 1–12.

[GS10] Gheyas I. A. y Smith L. S. (2010) A neural network-based framework for the reconstruction of incomplete data sets. *Neurocomputing* In Press.

[Han05] Han J. (2005) *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[HB02] Ho T. K. y Basu M. (2002) Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(3): 289–300.

[HB06] Ho T. K. y Basu M. (2006) *Data Complexity*. Springer-Verlag New York, Inc.

[INN04] Ishibuchi H., Nakashima T., y Nii M. (2004) *Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining*. Springer–Verlag.

[IY05] Ishibuchi H. y Yamamoto T. (2005) Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems* 13: 428–435.

[IYN05] Ishibuchi H., Yamamoto T., y Nakashima T. (2005) Hybridization of fuzzy GBML approaches for pattern classification problems. *IEEE Transactions on System, Man and Cybernetics B* 35(2): 359–365.

[Kal02] Kalousis A. (2002) *Algorithm selection via meta-learning*. PhD thesis, Université de Geneve.

[KC02a] Kwak N. y Choi C.-H. (2002) Input feature selection by mutual information based on parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(12): 1667–1671.

[KC02b] Kwak N. y Choi C.-H. (2002) Input feature selection for classification problems. *IEEE Transactions on Neural Networks* 13(1): 143–159.

[KCH+03] Kim W., Choi B.-J., Hong E.-K., Kim S.-K., y Lee D. (2003) A taxonomy of dirty data. *Data Mining and Knowledge Discovery* 7(1): 81–99.

[Kon05] Konar A. (2005) *Computational Intelligence: Principles, Techniques and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Kun00] Kuncheva L. (2000) *Fuzzy classifier design*. Springer, Berlin.

[lCvH92] le Cessie S. y van Houwelingen J. (1992) Ridge estimators in logistic regression. *Applied Statistics* 41(1): 191–201.

[LR87] Little R. J. A. y Rubin D. B. (1987) *Statistical Analysis with Missing Data*. Wiley Series in Probability and Statistics. Wiley, New York, 1st edition.

[MAC+92] Musavi M. T., Ahmed W., Chan K. H., Faris K. B., y Hummels D. M. (1992) On the training of radial basis function classifiers. *Neural Networks* 5(4): 595–603.

[Mam74] Mamdani E. (1974) Applications of fuzzy algorithm for control a simple dynamic plant. *Proceedings of the IEEE* 121(12): 1585–1588.

[McL04] McLachlan G. (2004) *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley and Sons.

[MML86] Michalksi R., , Mozetic I., y Lavrac N. (1986) The multipurpose incremental learning system aq15 and its testing application to three medical domains. In *5th INational Conference on Artificial Intelligence ((AAAI'86).)*, pp. 1041–1045.

[Mol93] Moller M. F. (1993) A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks* 6(4): 525–533.

[MPBM08] Matsubara E. T., Prati R. C., Batista G. E. A. P. A., y Monard M. C. (2008) Missing value imputation using a semi-supervised rank aggregation approach. In Zaverucha G. y da Costa A. C. P. L. (Eds.) *SBIA*, volumen 5249 of *Lecture Notes in Computer Science*, pp. 217–226. Springer.

[OPBM08] Orriols-Puig A. y Bernadó-Mansilla E. (2008) Evolutionary rule-based systems for imbalanced data sets. *Soft Computing* 13(3): 213–225.

[PA05] Pham D. T. y Afify A. A. (2005) Rules-6: a simple rule induction algorithm for supporting decision making. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pp. 2184–2189.

[PA06] Pham D. T. y Afify A. A. (2006) SRI: A scalable rule induction algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 220(4): 537–552.

[PBGC00] Pfahringer B., Bensusan H., y Giraud-Carrier C. G. (2000) Meta-learning by landmarking various learning algorithms. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 743–750. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Pla91] Platt J. (1991) A resource-allocating network for function interpolation. *Neural Computation* 3: 213–225.

[Pla98] Platt J. (1998) Fast training of support vector machines using sequential minimal optimization. In Schlkopf B., Burges C., y Smola A. (Eds.) *Advances in Kernel Methods – Support Vector Learning*, pp. 42–65. MIT Press, Cambridge, MA.

[PLD05] Peng H., Long F., y Ding C. (2005) Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(8): 1226–1238.

[Pyl99] Pyle D. (1999) *Data preparation for data mining.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Qui93] Quinlan J. R. (1993) *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning).* Morgan Kaufmann.

[Sch97] Schafer J. L. (1997) *Analysis of Incomplete Multivariate Data.* Chapman & Hall, London.

[SWK09] Sun Y., Wong A. K. C., y Kamel M. S. (2009) Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence* 23(4): 687–719.

[TS85] Takagi T. y Sugeno M. (1985) Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on Systems Man and Cybernetics* 15(1): 116–132.

[TSK05] Tan P.-N., Steinbach M., y Kumar V. (2005) *Introduction to Data Mining.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[WC87] Wong A. K. C. y Chiu D. K. Y. (1987) Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Trans. Pattern Anal. Mach. Intell.* 9(6): 796–805.

[WF05] Witten I. H. y Frank E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Wil72] Wilson D. (1972) Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems and Man and Cybernetics* 2(3): 408–421.

[WM97] Wolpert D. y Macready W. G. (1997) No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1): 67–82.

[WP03] Weiss G. M. y Provost F. J. (2003) Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research* 19: 315–354.

[WSF95] Wang R. Y., Storey V. C., y Firth C. P. (1995) A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering* 7: 623–640.

[Wu96] Wu X. (1996) *Knowledge acquisition from databases*. Ablex Publishing Corp., Norwood, NJ, USA.

[WU99] Wu X. y Urpani D. (1999) Induction by attribute elimination. *IEEE Transactions on Knowledge and Data Engineering* 11(5): 805–812.

[WW10] Wang H. y Wang S. (2010) Mining incomplete survey data through classification. *Knowledge and Information Systems* 24(2): 221–233.

[YSS97] Yingwei L., Sundararajan N., y Saratchandran P. (1997) A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation* 9: 461–478.

[YW06] Yang Q. y Wu X. (2006) 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making* 5(4): 597–604.

[ZW00] Zheng Z. y Webb G. I. (2000) Lazy learning of bayesian rules. *Machine Learning* 41(1): 53–84.

[ZZY03] Zhang S., Zhang C., y Yang Q. (2003) Data preparation for data mining. *Applied Artificial Intelligence* 17(5-6): 375–381.