Knowl Inf Syst (2015) 42:147–180
DOI 10.1007/s10115-013-0700-4

REGULAR PAPER

An automatic extraction method of the domains of competence for learning classifiers using data complexity measures

Julián Luengo · Francisco Herrera

Received: 20 January 2013 / Revised: 5 September 2013 / Accepted: 28 September 2013 /
Published online: 15 October 2013

Abstract The constant appearance of algorithms and problems in data mining makes impossible to know in advance whether the model will perform well or poorly until it is applied, which can be costly. It would be useful to have a procedure that indicates, prior to the application of the learning algorithm and without needing a comparison with other methods, whether the outcome will be good or bad using the information available in the data. In this work, we present an automatic extraction method to determine the domains of competence of a classifier using a set of data complexity measures proposed for the task of classification. These domains codify the characteristics of the problems that are suitable or not for it, relating the concepts of data geometrical structures that may be difficult and the final accuracy obtained by any classifier. In order to do so, this proposal uses 12 metrics of data complexity acting over a large benchmark of datasets in order to analyze the behavior patterns of the method, obtaining intervals of data complexity measures with good or bad performance. As a representative for classifiers to analyze the proposal, three classical but different algorithms are used: C4.5, SVM and K-NN. From these intervals, two simple rules that describe the good or bad behaviors of the classifiers mentioned each are obtained, allowing the user to characterize the response quality of the methods from a dataset's complexity. These two rules have been validated using fresh problems, showing that they are general and accurate. Thus, it can be established when the classifier will perform well or poorly prior to its application.

J. Luengo (✉)
Dept. of Civil Engineering, University of Burgos, 09006 Burgos, Spain
e-mail: jluengo@ubu.es

F. Herrera
Dept. of Computer Science and Artificial Intelligence, CITIC-University of Granada,
18071 Granada, Spain

F. Herrera
Faculty of Computing and Information Technology—North Jeddah,
King Abdulaziz University, Jeddah 21589, Saudi Arabia
e-mail: herrera@decsai.ugr.es

Keywords Classification · Data complexity · Domains of competence · C4.5 · Support vector machines · K-nearest neighbor

1 Introduction

The continuous development of data mining and Machine Learning techniques had led to a vast amount of algorithms and techniques suitable for a wide range of problems. Even focusing on a particular data mining task like classification, a great number of approaches and applications are continuously proposed, from hybridizations of rough sets and approaches for imbalanced data [36], tackling the curse of dimensionality [11,42], improving the combination of classifiers [20] and to use class decomposition strategies to alleviate the effects of noise [37], just to name a few. This variety of techniques and the new applications that continuously appear has made the identification of a competitive algorithm for a given problem a complex task. Since there is no best algorithm for all the possible tasks [44] when a new proposal is made, a comparison using an standard bunch of datasets is carried out in order to compare and to analyze it with respect to similar existing techniques [10,15].

At the same time, new real world problems appear continuously, and in very few cases, the users can know in advance whether the selected algorithm or technique will work appropriately or not, usually thanks to their experience and/or expert knowledge. In the classification task, efforts have been made to identify the most influential learning techniques [44] that usually work best, but new proposals and the variety of problems diminish their influence overtime. More recent paradigms such as the Meta Learning [8] try to fill this gap by means of cross comparing the performance of the methods in order to obtain the most suitable one. However, these paired approaches suffer from several problems [33] that have hindered their wide adoption.

In contrast to the aforementioned methodologies, other modern approaches work on the data level. Issues such as the generality of the data, the interrelationships 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 [4] applied to quantify such particular aspects of the problem, which are considered relevant to the classification task [17]. Studies of data complexity metrics applied to particular classification learning methods have recently spread [5,7,14,38,39] where they exploit such measures in order to gather information from the data useful to classification-related tasks.

Please note that the information provided by the data complexity measures (DCMs) is independent from any classifier applied afterward. Thus, it is possible to analyze the aspects of the data that are complicating the learning process of a particular classifier relating the performance value of the latter with the former associated DCM values. A seminal work using these precepts was carried out in [25] where a single fuzzy classifier was analyzed by a human-based process successfully obtaining the ranges of several DCMs in which such classifier behaves well or poorly, the so-called domains of competence. Using this *ad-hoc* procedure, the interrelationships of the domains of competence of nature-related classifiers were analyzed in [26], pointing out that similar classifiers suffer and benefit from the same characteristics of the data.

In this work, we propose an automatic extraction method for obtaining the domains of competence of any classifier by means of DCMs. This proposal is able to overcome the previously enumerated shortcomings as it does not depend on the cross comparisons between methods and it avoids the human bias when extracting the intervals. These domains of competence identify the datasets for which the classifier obtains a prominent good or bad performance

using only the information provided by the DCMs. We consider 12 DCMs for each dataset as proposed by Ho and Basu [17] where each one takes into account different geometrical properties of the data as the overlaps in feature values from different classes, separability of classes, measures of geometry, topology and density of manifolds. They enable the automatic extraction method to be capable of analyzing different data characteristics and relating them to the performance of the classifiers. As a final consequence, the user is able to predict the classifier's behavior prior to its application only using the characteristics of the data and to have in mind suitable parameter configurations or hints for improving it.

In order to analyze and test the automatic extraction method proposed, we consider three classifiers of different nature which have been comprehensively used in the literature. They are the C4.5 decision tree [35], a support vector machine (SVM) [34,43] and the K-nearest neighbor classifier [28]. An initial set of 340 binary classification datasets created from real world problems acts as a representative sample provided they supply enough samples of each DCM in order to extract the domains of competence of the three classifiers. An additional bunch of 1,383 datasets is used to validate the domains of competence obtained by the automatic extraction method.

Obtaining the classifiers' domains of competence by means of the automatic extraction method involves the following steps:

1. It obtains intervals that describe when the classifiers perform well or poorly according to the data complexity values using the initial 340 datasets.
2. One rule for each interval is formulated where some information and conclusions about the behavior of these methods can be stated.
3. The individual rules' antecedents are combined using a disjunctive normal form in order to improve their support and interpretability. Finally, two mutually exclusionary rules which discriminate the classifier's good or bad behavior each are obtained.

The intervals which describe the performance of the classifiers are based on the following average values:

- Accuracy ratio, considering the average interval test accuracy, and its difference to the average global test accuracy (across all the initial 340 datasets) with respect to a specified threshold.
- Detection of the overfitting, by means of the difference between the training accuracy and test accuracy ratio.

The final two rules obtained per classifier codify its domains of competence: one rule for those datasets with the characteristics suitable for it and another rule for those that yield poor accuracy. A careful analysis shows that while most of the DCMs are useful to the automatic extraction method to build them, the precise DCM application varies depending on the classifier. The generalization abilities of the domains of competence are validated using a fresh set of 1,383 datasets. The DCM values for each dataset are computed, and then, the domains of competence decide whether the dataset will yield a good accuracy or not for the classifier. This prediction has been compared to the real accuracy output, showing that the domains of competence automatically extracted provide an accurate and general description of the good and bad datasets for a given classifier. The software that implements the automatic extraction method, along with the algorithmic notation, datasets and results can be obtained from the website associated with this paper: http://sci2s.ugr.es/DC-automatic-method.

The rest of this paper is organized as follows. In Sect. 2, the considered complexity measures are introduced as well as the most recent literature on the topic. Section 3 defines the domains of competence used, their related approach in the literature and their motivation.

Section 4 describes the automatic extraction method proposed in this work. In Sect. 5, we summarize the experimental framework, in which we show the datasets used, the classifiers' details and their parameters. In Sect. 6, we include the experimental results obtained with the automatic extraction method and the domains of competence extracted, along with their analysis. Finally, in Sect. 7, the concluding remarks are provided.

2 Data complexity measures

The DCMs are a set of numeric indicators of particular data aspects that have been identified as difficult for the classifiers. This section is devoted to provide an overview of these measures. In the subsequent sections, we first present a short review of recent studies of data complexity metrics (Sect. 2.1), and then, we describe the measures of overlapping (Sect. 2.2), measures of separability of classes (Sect. 2.3) and measures of geometry (Sect. 2.4) used in our study.

2.1 Recent studies on data complexity

There are some methods used in classification, either learner or preprocessing techniques, which work well with particular datasets, while other techniques work better with different ones. This is due to the fact that each classification dataset has particular characteristics that define it. The generality of the data, the interrelationships among the variables and other factors are key for the results of such methods. These issues have been quantified by an emergent field by defining a set of measures related to particular sources of the problem on which the behavior of classification methods usually depends to [4].

A seminal work on data complexity is [17] in which some complexity measures for binary classification problems are proposed gathering metrics of three types: overlaps in feature values from different classes; separability of classes; and measures of geometry, topology and density of manifolds. Extensions can be also found in the literature, such as in the work of Singh [40], which offers a review of data complexity measures and proposes two new ones.

From these works, different authors attempt to address different data mining problems using these measures. For example, Baumgartner and Somorjai [5] define specialized measures for regularized linear classifiers. Other authors try to explain the behavior of learning algorithms using these measures, optimizing the decision tree creation in the binarization of datasets [23] or to analyze Fuzzy-UCS and the model obtained when applied to data streams [32]. The DCMs have reached other related fields, such as gene expression analysis in Bioinformatics [22,30]. Recently, the information provided by the DCMs has been used to stress the importance of a correct selection of the datasets that act as a test bed when comparing classifiers [27].

The research efforts in data complexity are currently focused on two fronts. The first aims to establish suitable problems for a given classification algorithm using only the data characteristics and thus determining their domains of competence. In this line of research recent publications, e.g., the works of Luengo and Herrera [25] and Bernado-Mansilla and Ho [7] provide a first insight into the determination of an individual classifier's domains of competence. In line with this, Sanchez et al. [29] study the effect of the data complexity on the nearest neighbor classifier. The relationships between the domains of competence of similar classifiers were analyzed by Luengo and Herrera [26], indicating that related classifiers benefit from common sources of complexity of the data.

Table 1 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	Nonlinearity of 1NN classifier
	T1	Fraction of points with associated adherence subsets retained
	T2	Average number of points per dimension

DCMs are increasingly used in order to characterize when a preprocessing stage will be beneficial to a subsequent classification algorithm in many challenging domains. García et al. [14] firstly analyzed the behavior of the evolutionary prototype selection strategy using one complexity measure based on overlapping. Further developments resulted in a characterization of when preprocessing is beneficial in imbalanced [24] or noisy data [38]. The DCMs can also be used as a part of the preparation step itself. As an example, we must remark the work of Dong [12] where a feature selection algorithm based on complexity measures is proposed.

This paper follows the first research line. It aims to characterize when a problem is suitable for a classifier solely regarding to the data properties using the information provided by the DCMs. Those regions of the DCMs that indicate an easy problem will be associated to a good performance of the classifier and viceversa. It can be expected that such metrics will enable us to know in advance whether a given classifier will be appropriate for the given dataset.

In this study, 12 metrics proposed by Ho and Basu [17] will be considered. In the following subsections, these measures, classified by their family, are briefly presented. For a deeper description of their characteristics, the reader may consult [17]. They are summarized in Table 1. In the following subsections, we briefly describe these 12 measures, classified by their respective types.

2.2 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 dataset 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 2 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

$$\begin{aligned} MINMAX_i &= MIN(max(f_i, C_1), max(f_i, C_2)) \\ MAXMIN_i &= MAX(min(f_i, C_1), min(f_i, C_2)) \\ MAXMAX_i &= MAX(max(f_i, C_1), max(f_i, C_2)) \\ MINMIN_i &= MIN(min(f_i, C_1), min(f_i, C_2)) \end{aligned}$$

F2 measures the amount of overlap of the bounding boxes of 2 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 2 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 [16], 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.

2.3 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 2 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 [41]. The method minimizes the sum of distances of error points to the separating hyperplane (subtracting a constant margin):

$$\begin{aligned} &\text{minimize} \quad \mathbf{a^t t} \\ &\text{subject to} \quad \mathbf{Z^t w} + \mathbf{t} \geq \mathbf{b} \\ &\mathbf{t} \geq \mathbf{0} \end{aligned}$$

where $\mathbf{a}$ and $\mathbf{b}$ 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 **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 dataset.

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 dataset 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 dataset is used as a measure.

N2: *ratio of average intra/interclass 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 intraclass distances to the sum of the interclass 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 dataset. 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.

2.4 Measures of geometry, topology and density of manifolds

These measures evaluate to what extent 2 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 [18] proposed a measure for the nonlinearity of a classifier with respect to a given dataset. 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 [21]. 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.

T2: *average number of points per dimension*. This is a simple ratio of the number of points in the dataset over the number of feature dimensions, i.e.,

$$T2 = \frac{m}{n},$$

where m is the number of examples in the dataset, and n is the number of attributes of the dataset.

3 Domains of competence of classifiers

This section is devoted to the introduction and description of the domains of competence of classifiers. The motivation behind the proposal is presented in Sect. 3.1, and the concept of the domains of competence of the classifiers used in this paper is given in Sect. 3.2.

3.1 Approaches on the classifier performance characterization

To determine whether a learning method will perform well or poorly before building and validating the model obtained, considering accuracy as the performance measure is not a trivial task. One of the best-known approaches to predict the classifier performance, which formalized this task [6,8,33], is the Meta Learning problem. Meta Learning intends 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 and 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 a Machine Learning problem instance was tackled using diverse descriptors, e.g., number of examples, number of attributes, percentage of missing values and landmarkers. [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 [19] indicates that the representativity of the problems and the perturbation induces strong biases in the Meta Learning classifier.

For these reasons, Meta Learning has achieved limited success.

We can also refer to the less known phase transition approach as a paradigm that aims to the same objective. Phase transition was initially developed to better understand the performances of Constraint Satisfaction algorithms indicating where the really hard problems are [9]. This paradigm defines a regular complexity landscape: 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.

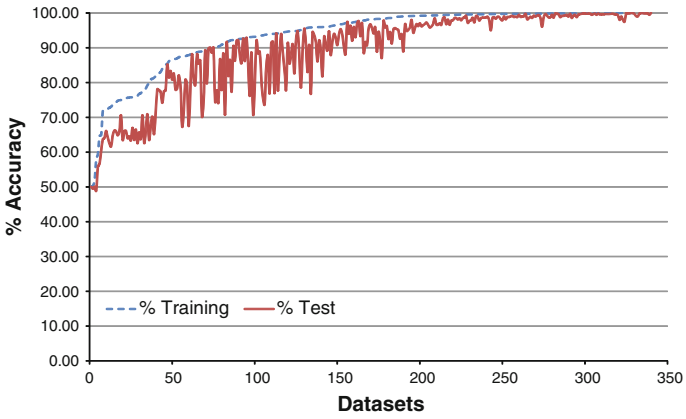

Fig. 1 Accuracy in training/test for C4.5 sorted by training accuracy

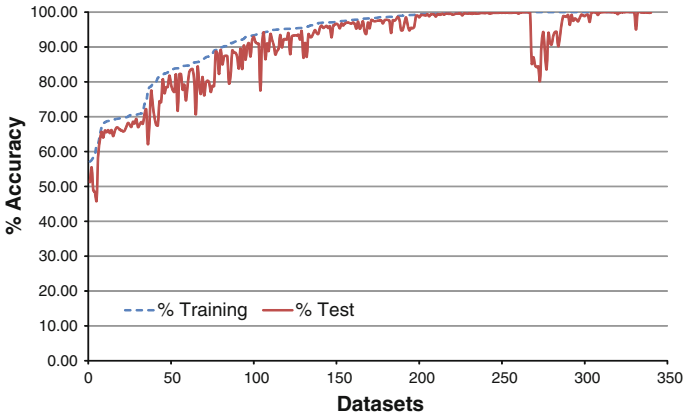

Fig. 2 Accuracy in training/test for SVM sorted by training accuracy

Using the phase transition, Baskiotis and Sebag [3] 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 datasets of different characteristics (number of attributes, number of terms, etc.) based on a uniform distribution. C4.5 is then trained on these datasets, and C4.5's error constitutes the complexity landscape (i.e., the competence map) using different datasets' 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.

One of the major problems with these approaches is that a good accuracy value in training does not necessarily mean that a good value in test will follow, neither similar training accuracy values will yield similar test accuracy values as well. Figures 1, 2 and 3 depict the accuracy results in training and test for C4.5, SVM and K-NN over all the initial 340 datasets used in this work (see Sect. 5.1), plotted in ascending training accuracy value. It is necessary to point out how overfitting is continuously present, specially for C4.5 and SVM. Therefore,

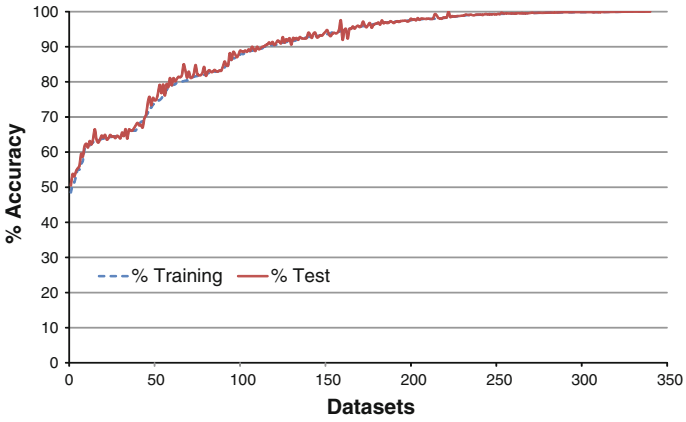

Fig. 3 Accuracy in training/test for K-NN sorted by training accuracy

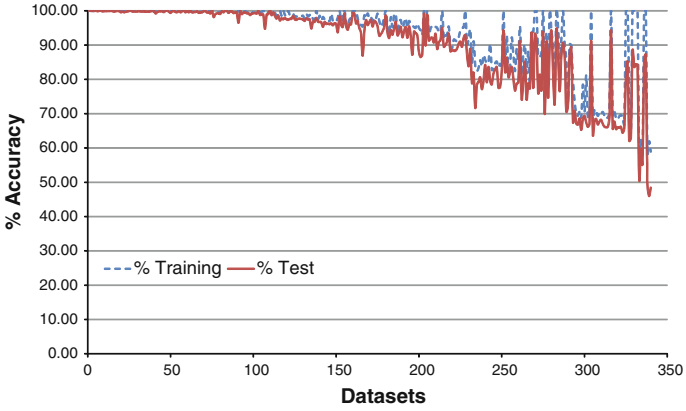

Fig. 4 Accuracy in training/test for SVM sorted by N1

the necessity of other kinds of approaches which do not use accuracy for characterizing the behavior of classifiers appears.

The DCMs presented in the previous section are a recent and promising option that can be used to establish the difficulty of a classification problem independently of the classifier. Each data complexity captures a singular aspect of the difficulty of the data, independently of the classifier applied afterward. Thus, the difficulty of the problem can be stated before the application of the classification learning. If the DCM identifies precisely, the source of difficulty for a given classifier and a given problem, any other dataset with a close value for the same DCM should produce a similar behavior by the classifier. For example, in Fig. 4, which shows SVM's accuracy sorted by the N1 DCM, datasets with close values for N1 show similar good or bad results. It can be expected that any other dataset with low N1 values will yield good accuracy values for SVM and poor accuracy values for high N1 values. However, this behavior pattern does not usually apply for all the DCMs. Figure 5 shows the same SVM's accuracy results sorted by the F2 DCM as an example of a DCM in which no significant regions can be found for either good or bad behavior.

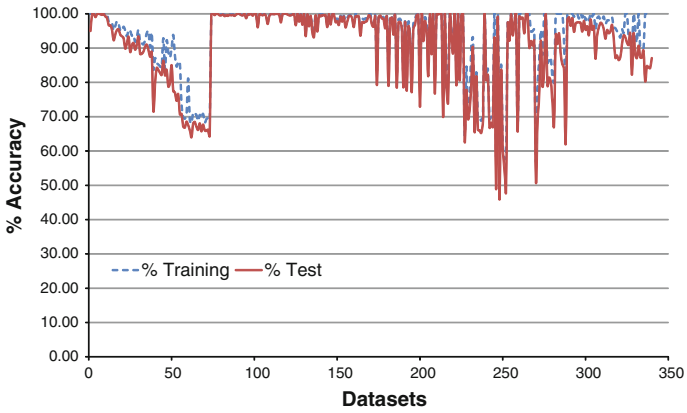

Fig. 5 Accuracy in training/test for SVM sorted by F2

Please note that any dataset may present diverse sources of difficulty for the classification task. It is needed to analyze several DCMs and their interactions in order to obtain a more complete picture of those easy or difficult problems for a classifier. This goal has been tackled by means of the domains of competence based on DCMs described in the following section.

3.2 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 based on several data properties at once. 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 DCMs. Such domains may discourage the use of a classification method if training it takes a long time, due to the inner complexity of the learning algorithm or the size of the data, and its performance is expected to be poor. Problems denoted as difficult *a priori* may also require a different parameter configuration with respect to simpler ones. The DCMs also give clues on the nature of the data as may be the importance of individual features, the shape of the class convex hulls or even the distance and/or separability of the examples at the class borders, making possible to focus specific efforts to improve the method's performance in the difficult regions.

Bernadó and Ho [7] introduced a first approach to the concept of domains of competence for the XCS classifier. Their initial definition indicates the region of the complexity space of problems adequate to the learning method characteristics. In order to establish such limits, they used six of the 12 DCMs presented in Table 1 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. Such a characterization could be useful for focusing the efforts of the learning algorithm's improvements on those difficult areas.

This concept of domains of competence can be related to other approaches in the literature presented in the previous section. The competence maps described by the phase transition paradigm can be considered as a limited version of these domains of competence. They suffer from severe limitations that have not been overcome: the real world datasets are rarely completely binary and uniformly distributed. The use of the domains of competence does not

suffer from these problems, as it is not dependent on a specific Machine Learning method either in the data distribution or the kind of dataset attributes.

In [25], the initial notion of domains of competence is extended and completed eliminating the constraints on the low or high values for the FH-GBML Fuzzy Rule-Based Classification System, using all of the 12 DCMs. Intervals of the measures with precise bounds in which the FH-GBML method performs well or poorly (instead of relating its performance to "high" or "low" values) are extracted ad-hoc. Using enough datasets sorted by a particular DCM, we can visually observe regions in which the method is performing noticeably well or poorly. These regions or intervals from several DCMs were coded as rules which constitute the domains of competence of the learning method.

It is desirable that the task of extracting the intervals in order to obtain the domains of competence is performed automatically. Orriols and Casillas [32] proposed the use of C4.5 to accomplish this task involving several classifiers at once. They used the data complexity values as the input features for each dataset and the label of the best algorithm for each problem as the output attribute resulting in a synthetic dataset. The rules obtained by C4.5 are used to code the intervals where each classifier outperforms the other ones. However, the domains of competence of two different classifiers cannot be compared in order to determine the best classifiers as showed in [32]: a better performance of one classifier does not mean that the other is performing badly as the absolute accuracy values were not being taken into account. Only shared strengths and weaknesses between classifiers can be pointed out as shown in [26].

In this paper, we extend and refine the methodology made in [25], proposing an automatic method in order to extract the domains of competence of an individual classifier, without the need to compare it to another one. An independent definition of the domains of competence will not vary with the consideration of new algorithms or problems. As stated before, the comparison between classifiers is directly related to the Meta Learning approach where many problems arise when trying to determine the best algorithm and no satisfactory solution has been proposed. Our proposal does not suffer from these issues, such as [32], while it has been proven that the obtained domains of competence are generalizable and extensible to other classifiers [26].

The use of the domains of competence instead of running the classifier over the problem has many advantages. For example, big datasets may cause the classifier to take too much time to train, or the adequate parameter configuration may not be obvious. Each DCM has a meaning, and hence, the causes for a classifier's easy or difficult problems can be identified. These issues can be addressed by means of the DCMs as they work directly with the data, but this information cannot be extracted from the Meta Learning or phase transition approaches.

4 Automatic extraction method

In the study performed in [25], an ad-hoc method used for extracting intervals for the FH-GBML was proposed. The intervals were extracted over the sorted datasets as described in Sect. 3.1. 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.

Two main problems were found when dealing with the ad-hoc process of extracting intervals in [25]:

1. The cut points which define the intervals were arbitrarily selected according to the graphics.

2. It is possible to omit similar intervals to the extracted ones. The user has no means to visually take into account all the eligible intervals.

In this work, we aim to solve or diminish the harmful effect of these problems. By proposing the automatic extraction method, we do not need to decide which DCMs are useful or not, as it will thoroughly search all the feasible intervals. These intervals are extracted based on a computer-understandable definitions equivalent to those performed by a human as in [25]. Please be aware that these definitions still need a certain degree of human interpretability of what is good and bad performance. However, under these premises, the automatic extraction method is able to extract the largest possible intervals that met the definitions, while it can drop out those small intervals that may constitute outliers.

In Sect. 4.1, the initial considerations for the definition of the good or bad behavior elements and intervals are provided. Sections 4.2 and 4.3 give the good and bad behavior elements and intervals definitions, respectively. Finally, the automatic extraction method is described in Sect. 4.4.

4.1 Initial considerations

The automatic extraction method analyzes a list of datasets where each complete dataset has an associated performance measure of the classifier considered (typically the training and test accuracy rates) and the values of the 12 DCMs.

Let $U = \{u_1, u_2, \ldots, u_n\}$ be a list of n different datasets. Each dataset 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 dataset u_i for a specific classifier, u_i^{tst} is the test accuracy value associated with the dataset u_i for the same classifier and the set $CM_{u_i} = \{u_i^{F1}, u_i^{F2}, \ldots, u_i^{T2}\}$ contains the values for the 12 DCMs.

Given a list of datasets $U = \{u_1, u_2, \ldots, u_n\}$, we denote 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 a classifier, intervals of values of the DCMs need to be stated. The intervals are defined over the list U of datasets sorting the list U by one DCM CM_j of the 12 DCMs obtaining a sorted list of datasets U_{CM_j} with respect to the data complexity CM_j 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 $i < j$.

Given a list of sorted datasets $U_{CM_j} = \{u_1, u_2, \ldots, u_n\}$ by the DCM $CM_j \in T$, 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 (datasets) and good and bad behavior intervals. The latter ones are obtained using the former ones, and they are described in the next subsections.

4.2 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 datasets, which

act as a sample of the classifier's behavior, so we need to parameterize these two subjective indicators.

The previous aspect is evaluated comparing the element's difference between training and test accuracy with respect to the global average. This global average difference is obtained as

$$\bar{U}^{diff} = \frac{1}{n} \sum_{j=1}^{n} max \left\{ u_j^{tra} - u_j^{tst}, 0 \right\}. \quad (1)$$

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 1 A **good behavior element** u_i is such that

1. $u_i^{test} \geq minGoodElementTest$; and
2. $u_i^{tra} - u_i^{tst} \leq \bar{U}^{diff}$.

Definition 2 A **bad behavior element** u_i is such that

1. $u_i^{test} < minGoodElementTest$; and
2. $u_i^{tra} - u_i^{tst} > \bar{U}^{diff}$.

Due to the first item of both definitions, no element can be considered as a good behavior and bad behavior element simultaneously.

4.3 Good and bad behavior intervals

The good or bad behavior intervals V show 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 in Eq. (1) limited to the datasets included in V:

$$\bar{V}^{diff} = \frac{1}{|V|} \sum_{u_j \in V}^{n} max \left\{ u_j^{tra} - u_j^{tst}, 0 \right\}.$$

A good interval must have a lower average difference than the global average $\bar{U}^{diff}$ defined in Eq. (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 obtained in the dataset sample.

In the case of good behavior intervals, we also establish that no element can have a test accuracy below $minGoodElementTest$ percent. The reason to do so 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 3 An **interval of good behavior** $V = \{u_i, \ldots, u_j\}$ is such that

1. $\bar{V}^{diff} \leq \bar{U}^{diff}$; and
2. $\bar{V}^{tst} \geq \bar{U}^{tst} + threshold$; and
3. $\forall u_j \in V; u_j^{tst} \geq minGoodElementTest$

Definition 4 An **interval of bad behavior** $V = \{u_i, \ldots, u_j\}$ is such that

1. $\bar{V}^{diff} > \bar{U}^{diff}$; and
2. $\bar{U}^{tst} < \bar{U}^{tst} - threshold$.

4.4 Automatic extraction method description

The automatic extraction method obtains a series of good or bad behavior intervals (as indicated in Definitions 3 and 4 respectively) from a list of datasets U. Each dataset $u_i \in U$ has the associated tuple T containing the training and test accuracy values for a particular classifier and its 12 data complexity values.

In order to extract the good and bad behavior intervals, the automatic extraction method follows a bottom-up process. It firstly arranges the datasets in U by sorting them with one of the DCMs 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 1 and 2, respectively), the exploration stops. Please note that the traversing order has no effect as we exhaustively look for single elements.

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 higher level phase is applied. It consist of a generalization step where overlapped or slightly separated intervals of the same type are merged and provided that the corresponding definition of a good or bad interval is maintained. A filtering process is also applied in order to avoid nonsignificant intervals where intervals with a support under 15 % of the total datasets are discarded.

We present the main outline of the automatic extraction method described in Algorithm 1.[1]

For the sake of brevity, the full algorithmic notation of the functions used in Algorithm 1 is provided in the associated web page. Their purpose is briefly 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.
- $nextImportantBadPoint(u_i, U)$: 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.

[1] The software that implements the automatic extraction method can be downloaded from the associated webpage.

Algorithm 1 Automatic Extraction Method

Input: A list of datasets $U = \{u_1, u_2, \ldots, u_n\}$. Each dataset u_i has associated a tuple T containing the training and test accuracy values for a particular learning method and its 12 data complexity values.
Output: A set of intervals G in which the learning method shows good 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}

- $extendGoodInterval(pos, U)$: 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 (datasets), provided that the new merged intervals satisfy Definition 3 and 4 of good or bad behavior, respectively. Please note that, this step may introduce some elements with accuracy values below $minGoodElementTest$ into the good behavior intervals.
- $dropSmallIntervals(A)$: This function discards the intervals $V_k \in A$, which contains a number of datasets less than $0.15 \cdot n$.

Table 2 Base datasets used to extract the domains of competence (extraction) and to validate them (validation)

Extraction	iris, wine, led7digit, ecoli, glass, tae, contraceptive, yeastB, segment, abalone, penbased, kr-vs-k
Validation	flare, zoo, balance, cars, mammographic, hayes-roth, australian, bupa, monks, new-thyroid, vowel, letter, marketing, texture

5 Experimental framework

In this section, we first describe the datasets considered for the domains of competence extraction and evaluation in Sect. 5.1. Next, we show the parameter values for the classification methods used in Sect. 5.2.

More detailed descriptions of the classifiers can be obtained from the associated webpage. All the datasets' generation details and their download packages are also available.

5.1 Datasets' choice for the experimental study

In this paper, a set of binary classification problems is used due to the fact that the DCMs are well defined only for two class problems. These problems were generated from pairwise combinations of the classes of 26 problems from the KEEL Dataset repository[2] [2] where datasets from well-known sources such as the UCI repository are stored and partitioned in KEEL format. We take each dataset and extract the examples belonging to each class, and a new dataset with the combination of the examples from two different classes is constructed. This will result in a new dataset with only 2 classes and the examples, which have two such classes as output. In order to obtain additional datasets, we also extend this methodology grouping the classes two by two, thus creating new class labels containing the former pair of classes.

It is important to note that not all the datasets generated by this procedure may be valid. In order to avoid the datasets that may yield erroneous conclusions, we filter those created datasets that resulted to be too easy for the classification task or present an imbalanced ratio between the classes. If the dataset proves to be linearly separable, then we may classify it with a linear classifier with no error and such a dataset would not be a representative problem. The complexity measure L1 indicates that if a problem is linearly separable when its value is zero, so every dataset with an L1 value of zero is discarded. On the other hand, we limit the imbalance ratio [31] to a value of 1.5, as it has been analyzed in the literature as a reasonable threshold to denote balanced datasets [13] that enable the use of accuracy as a performance measure (as used by the DCMs as well). This filtering resulted in 340 binary classification problems, which are used as our training bed for the automatic extraction method.

In order to validate the domains of competence obtained using the aforementioned 340 datasets, we have applied the same methodology to a different set of datasets from the KEEL dataset repository. Applying the same filtering criteria, we have obtained another 1,383 datasets used for validating the rules obtained in our analysis, which provide enough values for each DCM to be representative. Table 2 enumerates the base datasets used to obtain the extraction and validation datasets.

The learning methods' accuracy is estimated using a tenfold cross validation scheme where the datasets are partitioned in 10 equal-sized folds. Each time one fold acts as the test

[2] http://keel.es/datasets.php.

Table 3 Parameters used by the classifiers

Classifier	Reference	Parameters
C4.5	[35]	Confidence level = 0.25
		Pruning for the final tree = yes
SVM	[34]	C = [1, 1,000]
		Tolerance Parameter = 0.001
		Epsilon = 10^{-12}
		Kernel type = {Polynomial, RBF, Puk}
		ω = [0.0, 1.0]
		σ = [0.0, 1.0]
K-NN	[28]	K= {1,3,5,7,9,11,13}
		Distance measure = HVDM

Those parameters adjusted by grid search appear underlined

partition, while the rest are used to train and generate the model by the classifier. We take the average accuracy of training and test over the 10 partitions as a representative measure of the classifiers' performance. The DCMs were intended to reflect the difficulty with respect to accuracy, and the use of balanced dataset allows it usage without bias.

5.2 Parameters of the methods

In this section, we present the parameters used for the three classifiers considered using the values recommended by the authors. For those parameters that need to be empirically adjusted (in SVM and K-NN), a systematical grid search is applied over each one of the 340 initial datasets within the bounds indicated in the table in order to obtain the best test accuracy possible. We have used the implementations of these learning classifiers available in KEEL software [1]. The parameters are shown in Table 3, indicating the ranges of search if applicable.

6 Extracting the domains of competence using the automatic method

In this section, we present the analysis of the application of the automatic extraction method step by step in order to obtain the domains of competence of the three classifiers considered. First, we extract the intervals by using the automatic extraction method described in Sect. 4 for the three classifiers using the initial 340 datasets. These intervals and the rules obtained from them that provide a first approximation to the domains of competence of the three classifiers are shown in Sect. 6.1. The final and simple domains of competence description are tackled in Sect. 6.2 where the analysis of the disjunctive rules and their conjunctive combination is presented. Finally, these latter rules which represent the domains of competence of the classifiers are tested with the 1,383 validation datasets in Sect. 6.3.

6.1 Extracting the intervals and initial rules

In order to apply the automatic extraction method proposed in this paper, the threshold parameter needs to be provided. As mentioned before, this parameter establishes the degree of difference between the candidate intervals with respect to the average behavior of the method. In Table 4, we summarize the global training and test accuracy obtained by the

Table 4 Global average training and test accuracy/SD for C4.5, SVM and K-NN over the 340 datasets

	% Accuracy training & SD	% Accuracy test & SD
C4.5	93.79 % ± 9.89	89.32 % ± 12.61
SVM	93.21 % ± 11.23	90.30 % ± 12.47
K-NN	89.75 % ± 13.36	90.07 % ± 13.10

classifiers for the initial 340 datasets that will be used by the automatic extraction method as reference. All the detailed results can be downloaded from the associated webpage.

In Table 4, we also include the standard deviation of the accuracy of each classifier. It provides a hint on how consistent is the classifier over the 340 datasets. As can be appreciated, C4.5 and SVM present a lower standard deviation, while K-NN is less regular, but all three classifiers are comparable by the average test accuracy.

Before we can run the automatic extraction method, we need to set its parameters as described in Sect. 4. Please note that, the parameters could be adjusted to a particular scenario. That is, the concept of good or bad behavior varies according to the context, and it can be tuned depending on the user or the requirements needed. The selected values of the two parameters are $minGoodElementTest = 90$ and $threshold = 8$.

The *threshold* has been set high enough to produce a good differentiation between the intervals of good and bad behavior without exceeding the 100 % in test accuracy. A 90 % test accuracy is a reasonable borderline established by *minGoodElementTest* as can be seen from Table 4, forcing a good dataset to be classified at least as the average.

We run the automatic extraction method in order to obtain all the DCMs intervals in which each classifier has obtained significantly good or bad behavior with respect to the global performance. In Table 5, we summarize the intervals obtained by the automatic extraction method. We must point out that some of the intervals are shared among the classifiers, if not completely, at least in part of their supported datasets. It is also interesting to note how the automatic extraction method is capable of extracting distinct intervals for the different classifiers 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 a similar number of intervals for the three classifiers.

In overall terms, the intervals of good/bad behavior of the three classifiers appear in the same lower or higher valued regions of a DCM, but the support and bounds do not always coincide. The software which implements the automatic extraction method is also able to automatically plot the found intervals. Figures 6 and 7 graphically depict the intervals drawn by the software for SVM and C4.5, respectively, considering the F3 DCM. They show an example of such a difference in behavior reflected by the distinct intervals yielded by each classifier. While SVM is able to attain similar accuracy in close regions of higher values of F3 DCM, C4.5 shows a lesser identifiable behavior pattern, thus hindering the support of the obtained intervals. Some datasets that lie in the bad behavior interval show a good performance. This may happen because some datasets may appear to be difficult from one DCM perspective, but they may be easy to classify considering other aspects of the data (other DCMs).

From the intervals, a set of rules is created automatically by the software that will act as the germinal description of the domains of competence of the classifier for each DCM. Table 6 depicts the rules for C4.5, Table 7 shows the rules for SVM and Table 8 contains the rules for K-NN with the support of the rules over the 340 datasets, the average training and test accuracy of the covered datasets and the difference with the global accuracy shown in Table 4,

Table 5 Intervals obtained by the automatic extraction method for the 340 datasets. The MAX value indicates that the interval extends to the maximum value of the measure that can be either 1 or ∞

C4.5				SVM				K-NN			
Good behavior		Bad behavior		Good behavior		Bad behavior		Good behavior		Bad behavior	
Measure	Interval	Measure	Interval	Measure	Interval	Measure	Interval	Measure	Interval	Measure	Interval
F3	[0.7523,MAX]	F1	[0.02005,1.204]	F3	[1.0,MAX]	F1	[0.02005,1.093]	F3	[0.8271,MAX]	F1	[0.0,1.001]
N1	[0.0,0.145]	F3	[0.00675,0.7508]	N1	[0.0,0.12]	F2	[0.02686,0.75]	N1	[0.0,0.1529]	F2	[0.02763,0.75]
N2	[0.0,0.3106]	N1	[0.3165,MAX]	N2	[0.0,0.1897]	F3	[0.00941,0.6803]	N2	[0.0,0.2305]	F3	[0.00675,0.6121]
N3	[0.0,0.09234]	N2	[0.6227,MAX]	N3	[0.0,0.07299]	N1	[0.2569,MAX]	N3	[0.0,0.06667]	N1	[0.401,MAX]
L1	[0.3267,0.549]	N3	[0.1846,MAX]	N4	[0.0,0.04481]	N2	[0.6099,MAX]	N4	[0.0,0.04717]	N2	[0.7149,MAX]
L2	[0.0,0.09773]	N4	[0.2976,MAX]	L1	[0.316,0.5211]	N3	[0.1743,MAX]	L1	[0.316,0.5172]	N3	[0.2555,MAX]
L3	[0.00544,0.06439]	L1	[0.8093,MAX]	L2	[0.0,0.08106]	N4	[0.2982,MAX]	L2	[0.0,0.07197]	N4	[0.3094,MAX]
T1	[0.0,0.9705]	L2	[0.2889,MAX]	L3	[0.00544,0.02045]	L1	[0.8093,3.104]	L3	[0.0,0.02251]	L1	[0.8453,11.04]
T2	[65.62,74.0]	L3	[0.2552,MAX]	T1	[0.0,0.9536]	L2	[0.3377,MAX]	T1	[0.0,0.9536]	L3	[0.487,MAX]
		T1	[0.9919,MAX]	T2	[65.62,74.0]	L3	[0.2513,MAX]	T2	[65.62,82.0]	T1	[0.9948,MAX]
		T2	[82.0,MAX]							T2	[109.6,1165.0]

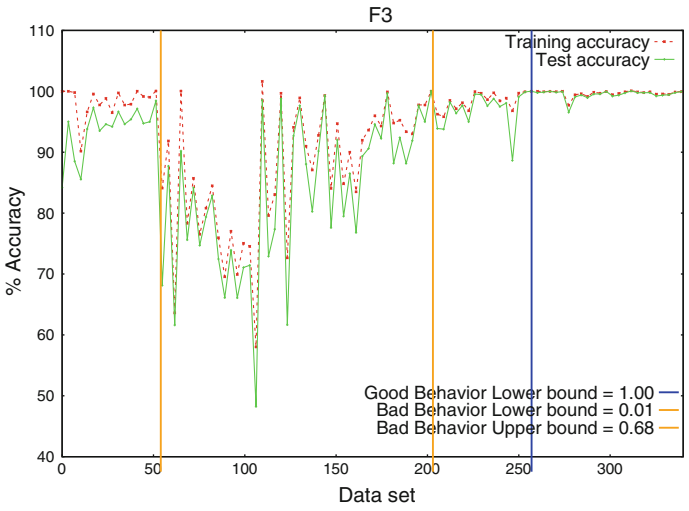

Fig. 6 SVM accuracy results sorted by F3 with the extracted intervals

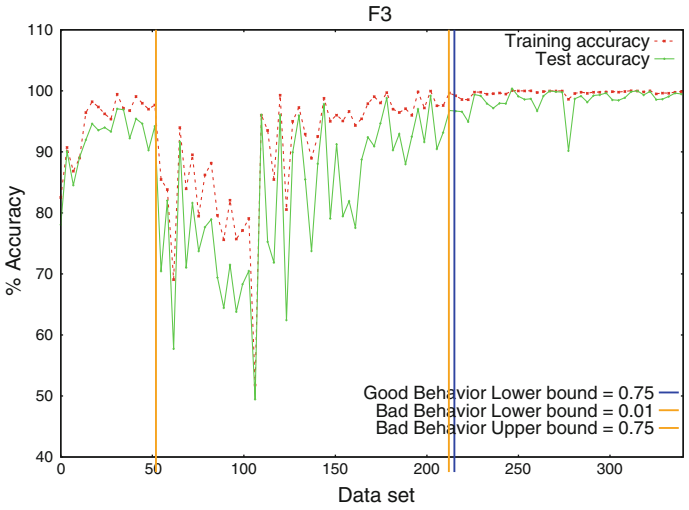

Fig. 7 C4.5 accuracy results sorted by F3 with the extracted intervals

The columns in Tables 6, 7 and 8 are organized as follows:

- The first column corresponds to the identifier of the rule.
- The "Range" column presents the domain of the rule.
- The third column "Support" presents the percentage of datasets, which verify the antecedent part of the rule.
- The column "% Training, SD" 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 of the rule and the average training accuracy.

Table 6 Rules obtained for C4.5 from the automatic intervals

Id.	Range	Support (%)	Training accuracy (%)	Training difference	Test accuracy (%)	Test difference
Good behavior rules						
R1+	F3∈[0.7523,MAX]	36.76	99.73	5.94	98.72	9.40
R2+	N1∈[0.0,0.145]	55.88	99.20	5.41	97.48	8.15
R3+	N2∈[0.0,0.3106]	53.24	98.88	5.09	97.36	8.04
R4+	N3∈[0.0,0.09234]	57.06	99.17	5.39	97.33	8.01
R5+	L1∈[0.3267,0.549]	24.41	98.87	5.08	97.33	8.00
R6+	L2∈[0.0,0.09773]	46.47	98.86	5.08	97.37	8.05
R7+	L3∈[0.00544,0.06439]	33.53	98.77	4.99	97.40	8.08
R8+	T1∈[0.0,0.9705]	45.88	99.24	5.46	97.54	8.21
R9+	T2∈[65.62,74.0]	31.18	99.62	5.83	98.43	9.11
Bad behavior rules						
R1−	F1∈[0.02005,1.204]	40.29	89.04	−4.75	81.31	−8.02
R2−	F3∈[0.00675,0.7508]	47.35	89.05	−4.74	81.31	−8.01
R3−	N1∈[0.3165,MAX]	31.76	83.71	−10.08	74.92	−14.41
R4−	N2∈[0.6227,MAX]	25.88	82.41	−11.38	73.82	−15.50
R5−	N3∈[0.1846,MAX]	30.29	83.24	−10.54	74.51	−14.81
R6−	N4∈[0.2976,MAX]	20.59	80.49	−13.30	73.74	−15.58
R7−	L1∈[0.8093,MAX]	29.12	86.63	−7.16	79.32	−10.00
R8−	L2∈[0.2889,MAX]	28.53	83.76	−10.03	73.95	−15.37
R9−	L3∈[0.2552,MAX]	30.00	84.14	−9.65	74.56	−14.76
R10−	T1∈[0.9919,MAX]	39.41	87.26	−6.53	81.18	−8.14
R11−	T2∈[82.0,MAX]	24.71	86.10	−7.68	80.49	−8.84

- The column "% Test, SD" 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 average test accuracy.

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 datasets and obtain significant differences in accuracy greater and equal to the *threshold* used as parameter by the automatic extraction method. The differences for the good behavior rules in test are very close to such *threshold* parameter value given in the case of C4.5 and SVM. This is not the case for all the bad behavior rules, where the differences are usually greater than *threshold*. 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 extraction method uses the N1, N3, L1 and L2 DCMs in order to characterize the good datasets for the three classifiers. These measures deal with the percentage of examples in the class boundaries and their separation, and constitute

Table 7 Rules obtained for SVM from the automatic intervals

Id.	Range	Support (%)	Training accuracy (%)	Training difference	Test accuracy (%)	Test difference
Good behavior rules						
R1+	F3∈[1.0,MAX]	24.41	99.78	6.57	99.59	9.29
R2+	N1∈[0.0,0.12]	54.12	99.13	5.92	98.30	8.00
R3+	N2∈[0.0,0.1897]	41.18	99.25	6.04	98.34	8.04
R4+	N3∈[0.0,0.07299]	53.53	99.10	5.89	98.30	8.00
R5+	N4∈[0.0,0.04481]	25.00	99.49	6.28	98.41	8.12
R6+	L1∈[0.316,0.5211]	20.88	98.95	5.73	98.43	8.13
R7+	L2∈[0.0,0.08106]	42.35	99.31	6.10	98.33	8.03
R8+	L3∈[0.00544,0.02045]	16.76	99.36	6.15	98.48	8.19
R9+	T1∈[0.0,0.9536]	37.35	99.16	5.95	98.63	8.34
R10+	T2∈[65.62,74.0]	31.18	99.00	5.79	98.52	8.22
Bad behavior rules						
R1−	F1∈[0.02005,1.093]	38.53	86.57	−6.64	82.28	−8.01
R2−	F2∈[0.02686,0.75]	19.12	85.97	−7.24	81.19	−9.11
R3−	F3∈[0.00941,0.6803]	44.12	86.26	−6.95	82.13	−8.17
R4−	N1∈[0.2569,MAX]	32.35	82.62	−10.60	76.38	−13.91
R5−	N2∈[0.6099,MAX]	26.76	82.27	−10.95	76.20	−14.09
R6−	N3∈[0.1743,MAX]	31.47	82.72	−10.49	76.42	−13.88
R7−	N4∈[0.2982,MAX]	20.00	81.04	−12.17	74.98	−15.31
R8−	L1∈[0.8093,3.104]	23.82	81.92	−11.29	76.53	−13.76
R9−	L2∈[0.3377,MAX]	24.12	80.74	−12.47	74.21	−16.08
R10−	L3∈[0.2513,MAX]	30.29	82.78	−10.44	76.33	−13.96

the core of the measures of separability of classes, indicating that it is a key aspect to their performance. Apart from this core of measures, the automatic extraction method picks an extra set of measure for each classifier due to their different nature complementing the aforementioned four measures. As stated in [26], it can be expected that using more similar methods will produce greater coincidences in the extracted intervals. SVM and K-NN appear to share more of the complexity space in which they perform well than C4.5, whose good behavior can be described with less DCMs, indicating that it is less sensitive to geometrical properties in the data that may benefit it.

The intervals for the classifiers' bad behavior use approximately the same number of measures for each one, showing that the sources of difficulty of the data affect roughly similarly the three classifiers. However, while C4.5 and SVM bad performance can be described by fewer and larger intervals, K-NN shows prominent bad behavior for more and smaller regions. The less consistent performance of K-NN isolates small regions of the complexity space where the requirements are met.

With these simple and individual rules, an initial characterization of the datasets for which good or bad behavior is obtained for the classifiers can be considered. These individual rules can be used to describe the domains of competence of C4.5, SVM and K-NN, but some drawbacks can be pointed out:

Table 8 Rules obtained for K-NN from the automatic intervals

Id.	Range	Support (%)	Training accuracy (%)	Training difference	Test accuracy (%)	Test difference
Good behavior rules						
R1+	F3∈[0.8271,MAX]	35.29	99.07	9.32	99.15	9.07
R2+	N1∈[0.0,0.1529]	57.06	98.14	8.39	98.18	8.11
R3+	N2∈[0.0,0.2305]	47.94	98.22	8.47	98.28	8.2
R4+	N3∈[0.0,0.06667]	52.06	98.41	8.66	98.48	8.41
R5+	N4∈[0.0,0.04717]	25.88	97.98	8.23	98.11	8.04
R6+	L1∈[0.316,0.5172]	20.29	98.48	8.73	98.53	8.46
R7+	L2∈[0.0,0.07197]	40	98.17	8.42	98.24	8.17
R8+	L3∈[0.0,0.02251]	31.47	98.21	8.46	98.32	8.25
R9+	T1∈[0.0,0.9536]	37.35	98.63	8.88	98.69	8.61
R10+	T2∈[65.62,82.0]	31.47	98.65	8.9	98.65	8.57
Bad behavior rules						
R1−	F1∈[0.0,1.001]	39.41	81.48	−8.27	81.94	−8.14
R2−	F2∈[0.02763,0.75]	18.82	80.86	−8.89	81.27	−8.80
R3−	F3∈[0.00675,0.6121]	42.94	81.49	−8.26	81.99	−8.09
R4−	N1∈[0.401,MAX]	22.35	71.61	−18.14	72.22	−17.86
R5−	N2∈[0.7149,MAX]	20.29	74.77	−14.98	75.07	−15.01
R6−	N3∈[0.2555,MAX]	20.88	71.28	−18.47	71.78	−18.30
R7−	N4∈[0.3094,MAX]	19.12	73.09	−16.66	73.39	−16.69
R8−	L1∈[0.8453,11.04]	23.24	81.40	−8.35	81.91	−8.16
R9−	L3∈[0.487,MAX]	19.12	73.23	−16.52	74.13	−15.94
R10−	T1∈[0.9948,MAX]	36.76	81.58	−8.17	81.92	−8.15
R11−	T2∈[109.6,1165.0]	21.76	81.48	−8.27	81.61	−8.46

- It is very likely that good or bad datasets for the classifiers are identified each one as so by different rules, that is, different rules are capturing the same datasets as good/bad for the classifier constituting a redundant description of the domains of competence.
- It is possible that the application of the rules to a new dataset yields both *good* and *bad behavior* output. This case is presented when such a new dataset is covered by good and bad behavior rules simultaneously for two different DCMs.

Due to these reasons, a simpler and univocal representation of the domains of competence is desired. To do so, we consider the grouping and combination of the different rules in the following section.

6.2 Combination of the individual rules

The objective of this section is to analyze the effect of combining the rules to overcome the limitations presented by the description provided by their individual use. 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 rules will be

activated if any of the component rules' antecedents are verified using the disjunctive normal form. By means of merging the individual rules, we can arrive at a more general description, with a wider support, of the behavior of the classifiers' methods. PRD and NRD rules avoid the redundancy of the single rules as they group all the good or bad sources of complexity for a given dataset in a single rule.

As stated previously, the PRD and NRD rules may present overlapping in their support, but a mutually exclusive description of the good and bad regions is desirable [25]. In order to tackle this issue, we consider the conjunctive operator *and* ($\wedge$) and the difference operator *and not* ($\wedge\neg$) between the PRD and NRD rules. The difference will remove the datasets for which the classifiers present good or bad behavior from the disjunctive negative or positive rules, respectively. That is, by means of the difference, we intend to remove the datasets 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 datasets.

In Table 9, we depict the new collective rules for the classifiers. From them, we can point out the following for the two classifiers:

- The Positive Rule Disjunction (PRD) offers a higher support than the single rules for all the three classifiers. It also gives a good training and test accuracy comparable to the threshold value.
- The Negative Rule Disjunction (NRD) obtains a wider support as well. 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. That means that the individual rules that shape NRD usually contain datasets with good behavior that hinder the test accuracy difference as they sum up.
- The Positive and Negative Rule Disjunction (PRD$\wedge$NRD) is similar to PRD in the training and test accuracy difference, and it represents the good dataset for the classifiers covered by the rules of bad behavior that decrease the test difference in NRD. It roughly represents one third of the datasets contained in PRD.
- The Positive and Not Negative Rule Disjunction (PRD$\wedge\neg$NRD) has a lower support than PRD$\wedge$NRD, but its difference is higher than PRD and PRD$\wedge$NRD rules, since the datasets with low accuracy for the classifiers present in NRD have been removed in PRD$\wedge$NRD.
- The Negative and Not Positive Rule Disjunction (NRD$\wedge\neg$PRD) is a good rule to describe the bad behavior of the classifiers. It has a high support and a high difference in both training and test sets. When removing the good datasets of PRD$\wedge$NRD, the NRD$\wedge\neg$PRD rule becomes more accurate in the description of the bad datasets for the classifier.
- Only C4.5 has datasets not characterized by either the PRD rule or the NRD, and it presents a small support and a small difference from the global accuracy both in training and test.

From all the disjunctive and new conjunctive rules, we can present PRD as a representative description of good datasets and NRD$\wedge\neg$PRD as a representative description for bad datasets when using the classifiers. While a DCM only analyzes one difficulty source at once, it may happen that a difficult dataset for one DCM is actually an easy problem from the point of view of other(s) DCMs. A problem will be usually easy if one of the DCMs used in the domains of competence of the method indicates so. Thus, we can rely in the PRD definition with a higher degree of confidence than NRD: if a dataset is covered by PRD, it will be surely a good behavior problem for the classifier.

Table 9 Disjunctive rules from all simple rules for C4.5, SVM and K-NN

Id.	Range	Support (%)	Training accuracy (%)	Training difference	Test accuracy (%)	Test difference
C4.5						
PRD	If R1+ or …or R9+ then good behavior	67.35	98.45	4.67	96.27	6.95
NRD	If R1− or …or R11−then bad behavior	77.65	92.08	−1.71	86.58	−2.75
PRD ∧NRD	If PRD and NRD then good behavior	45.00	97.82	4.04	94.98	5.65
PRD ∧¬ NRD	If PRD and not NRD then good behavior	22.35	99.73	5.94	98.87	9.55
NRD∧¬PRD	If NRD and not PRD then bad behavior	32.65	84.15	−9.63	74.99	−14.33
Not characterized	If not PRD and not (NRD∧¬PRD) then uknown behavior	0.00	–	–	–	–
SVM						
PRD	If R1+ or …or R10+ then good behavior	62.94	98.61	5.39	97.46	7.16
NRD	If R1− or …or R10−then bad behavior	65.29	90.02	−3.20	86.14	−4.16
PRD ∧NRD	If PRD and NRD then good behavior	28.53	97.82	4.60	96.56	6.26
PRD ∧¬ NRD	If PRD and not NRD then good behavior	34.41	99.26	6.05	98.21	7.91
NRD∧¬PRD	If NRD and not PRD then bad behavior	36.76	83.96	−9.25	78.05	−12.24
Not characterized	If not PRD and not (NRD∧¬PRD) then uknown behavior	0.29	94.93	1.72	87.83	−2.47
K-NN						
PRD	If R1+ or …or R10+ then good behavior	63.24	97.17	7.42	97.28	7.21
NRD	If R1− or …or R11− then bad behavior	74.12	86.48	−3.27	86.92	−3.16
PRD ∧NRD	If PRD and NRD then good behavior	37.35	95.82	6.07	96.02	5.94

Table 9 continued

Id.	Range	Support (%)	Training accuracy (%)	Training difference	Test accuracy (%)	Test difference
PRD ∧¬ NRD	If PRD and not NRD then good behavior	25.88	99.12	9.37	99.11	9.04
NRD∧¬PRD	If NRD and not PRD then bad behavior	36.76	76.98	−12.77	77.67	−12.40
Not characterized	If not PRD and not (NRD∧¬PRD) then unknown behavior	0.00	–	–	–	–

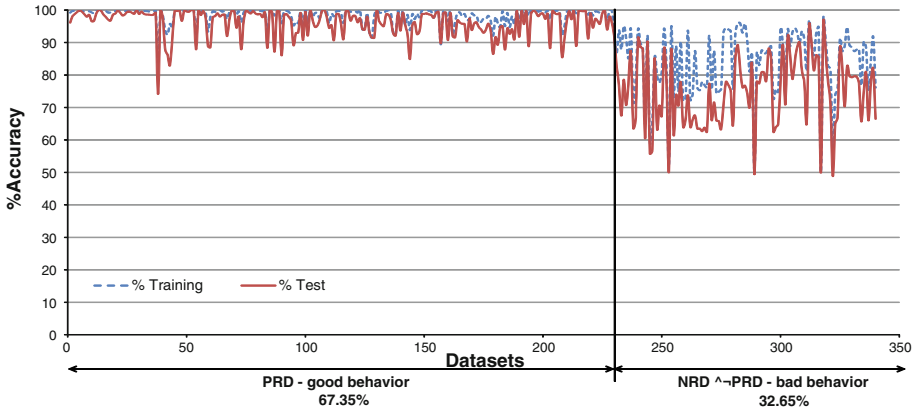

Fig. 8 C4.5 block representation for PRD, NRD∧¬PRD and not characterized datasets

One main advantage is that these rules are mutually exclusive providing an unique description for an unseen dataset. We can consider a representation of the 340 datasets in three blocks datasets with their respective support. In Figs. 8, 9 and 10, we depict the three block regions for the C4.5, SVM and K-NN, respectively. On the left block, the datasets covered by PRD are depicted. On the center block, the datasets covered by NRD∧¬PRD are plotted. The small region in the right corresponds to the not characterized datasets if any.

From Figs. 8, 9 and 10, we can observe that the 100 % of the analyzed datasets are characterized except for SVM with a 99 % of coverage. Using the ad-hoc extraction method in [25], the equivalent PRD and NRD∧¬PRD rules only covered 75 % of the datasets 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 datasets using the DCMs, as it is capable of overcoming the limitations derived from a human-driven procedure. The final description of the domains of competence of *good behavior* and *bad behavior* characterization for each classifier can be achieved by the PRD and NRD∧¬PRD rules.

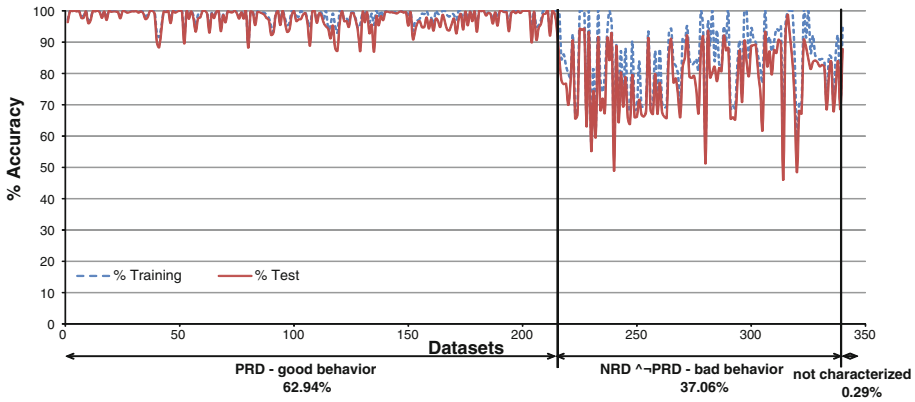

Fig. 9 SVM block representation for PRD, NRD∧¬PRD and not characterized datasets

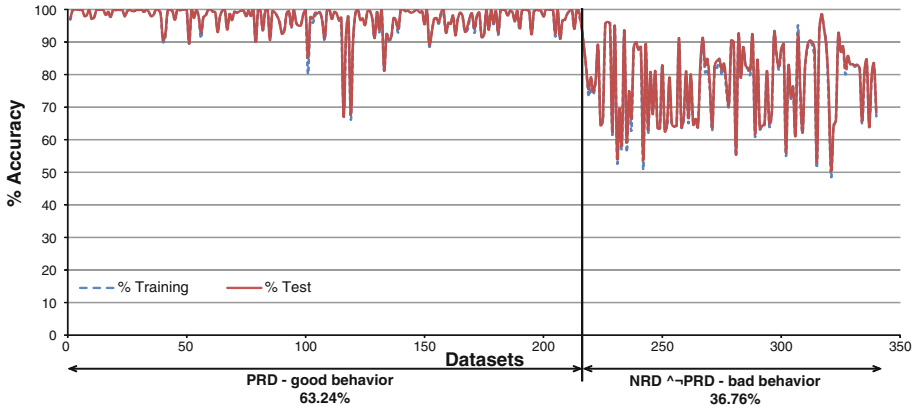

Fig. 10 K-NN block representation for PRD, NRD∧¬PRD and not characterized datasets

6.3 Validation of the domains of competence

Once we have obtained a set of descriptive rules of the domains of competence for the three classifiers with the automatic extraction method, their domains of competence have been established. They provide the description of the datasets that result easy or difficult for their learning phase both descriptively, by means of the DCMs involved, and quantitatively, by means of the rules presented in the previous section.

The domains of competence have little use if they do not yield comparable results with unseen datasets that were not used in the extraction step by the automatic extraction method. In order to validate and evaluate how well these domains of competence generalize for new cases, we use an extra bunch of datasets, which have not been considered previously.

As described in Sect. 5.1, we validate the PRD and NRD∧¬PRD rules using a set of 1,383 fresh datasets. In Table 10, we summarize the average training and test accuracy values for each classifier. As it can be seen, the test accuracy values are comparable to those obtained in Table 4, while the standard deviation is higher due to the large number of datasets used to validate.

Table 10 Global average training and test accuracy/SD for C4.5, SVM and K-NN over the validation datasets

	% Accuracy training & SD	% Accuracy test & SD
C4.5	95.20 % ± 8.36	90.94 % ± 14.10
SVM	95.37 % ± 8.47	92.26 % ± 14.06
K-NN	91.80 % ± 14.76	91.86 % ± 14.68

Table 11 Validation results for PRD and NRD∧¬PRD rules

Id.	Support (%)	Training accuracy (%)	Training difference	Test accuracy (%)	Test difference
C4.5					
PRD	77.58	99.57	4.24	98.11	7.19
NRD-PRD	22.42	80.66	−14.67	66.02	−24.89
Not characterized	0.00	–	–	–	–
SVM					
PRD	77.30	99.91	4.16	99.77	7.35
NRD-PRD	22.70	81.58	−14.17	67.40	−25.03
Not characterized	0.00	–	–	–	–
K−NN					
PRD	77.66	99.43	7.56	99.44	7.51
NRD-PRD	22.34	65.60	−26.27	65.81	−26.12
Not characterized	0.00	–	–	–	–

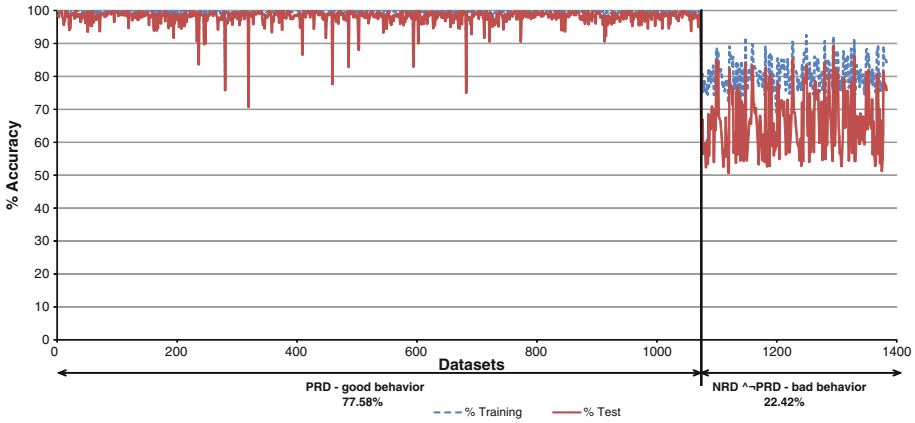

Fig. 11 C4.5 block representation for PRD and NRD∧¬PRD for validation datasets

Using the software to validate the intervals, we present Table 11 with the average training and test accuracy of the dataset covered by the PRD and NRD∧¬PRD rules for each classifier and the difference from the global average of the new 1,383 datasets. The data showed in Table 11 are plotted as a three blocks representation in Figs. 11, 12 and 13 and in a similar fashion to those depicted in Figs. 8, 9 and 10, respectively, to better visualize the validation results.

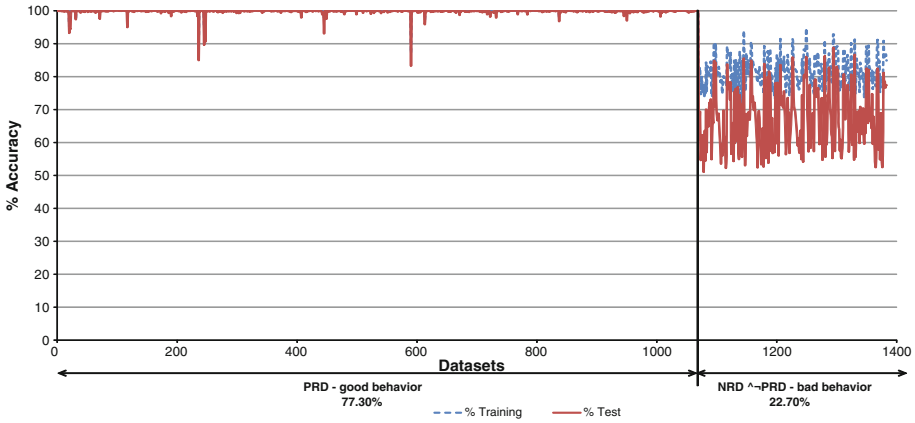

Fig. 12 SVM block representation for PRD and NRD∧¬PRD for validation datasets

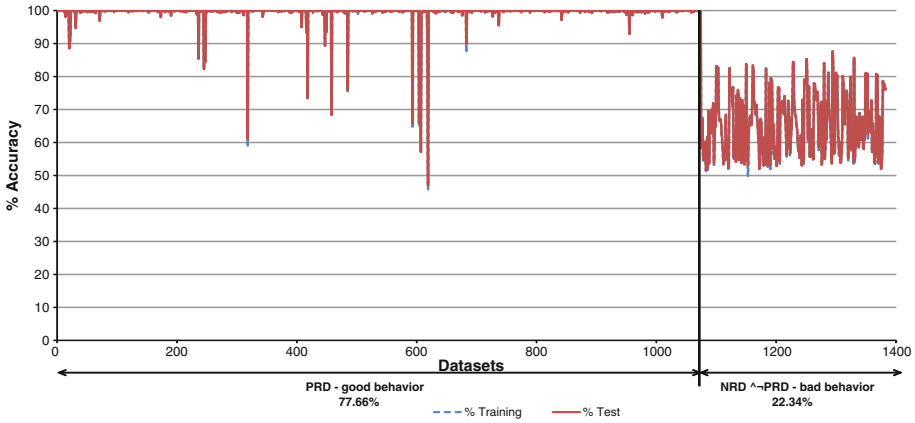

Fig. 13 K-NN block representation for PRD and NRD∧¬PRD for validation datasets

From these results, we can observe the following:

- It can be appreciated that the PRD rule covers the datasets for which the classifiers attain an outstanding accuracy, while the NRD∧¬PRD takes the datasets that verify the opposite. Only a few outliers are identified as good while presenting a poor accuracy in a ration of 10–1,000.
- The differences in test accuracy for the three classifiers in training and test are similar to those showed by the rules PRD and NRD∧¬PRD in Table 9 that act as a threshold for the domains of competence differentiation, even using 4 times more datasets.
- All the validation datasets are characterized showing the good generalization of the domains of competence obtained.

For the classifiers considered, their particular PRD and NRD∧¬PRD rules have a support of approximately 75 % and 25 % each, which accurately represent the balance of the good and bad datasets used for validation. It is reasonable to expect that most unseen datasets can be classified by the domains of competence. It is also important to stress that the definitions of good and bad behavior intervals given in Sect. 4 are independent of the classifier. Thus, the

PRD and NRD$\wedge\neg$PRD rules obtained from the use of the automatic extraction method can predict the behavior of any classifier by defining its domains of competence.

7 Concluding remarks

In the present paper, we have proposed a new automatic extraction method to obtain the domains of competence of a classifier, which allows to predict if any dataset will be suitable for such learning method or not. This automatic extraction method uses a set of data complexity metrics, which measure characteristics of the data independently of the learning method, enabling it to be applied over any classifier.

In order to check the validity of the proposal, we have performed a thorough study over a large set of binary datasets with three classical classifiers very different in their nature, and so, we can confirm the capabilities of this proposal: C4.5, SVM and K-NN. First, we have computed 12 data complexity measures for each dataset. We have used the automatic extraction method providing a significant margin with respect to the average classifier's behavior, to identify the good and bad problems for each classifier and to obtain a different set of intervals based on the information provided by the data complexity measures. Then, we have built descriptive rules from these intervals, and we have studied the interaction between them obtaining as a final result two rules which codify the domains of competence of the classifier. These rules describe the characteristics of the datasets that belong or not to the "sweet spot" of one classifier independently from the others, unlike previous proposals in the literature where cross comparisons are needed.

We have validated these rules with a large extra benchmark of datasets in order to check their generalization and prediction capabilities of the pair of rules that conform the domains of competence of each classifier. The results show that from a reduced bunch of datasets, around three hundred samples, the obtained domains of competence are general enough to identify the datasets suitable for the classifier maintaining the margin given to the automatic extraction method. Thanks to this proposal, we present the possibility of determining automatically which datasets will prove to be good or bad for the classifiers in advance to their execution using the data complexity measures once the domains of competence of the classifier have been established.

We must point out that this is a study that uses three specific methods as a representative set and can be extended as long as the data complexity measures for a dataset can be calculated. 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 classifiers for data mining. New questions such as to analyze the interpretation/connection between datasets in the same expertise domain, applying this proposal in real world datasets or the minimum amount of datasets required to establish the intervals also arise and will be tackled in future research.

Acknowledgments Supported by the Research Projects TIN2011-28488 and P10-TIC-06858.

References

1. 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 (2008) Keel: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13(3):307–318

2. Alcalá-Fdez Jesús, Fernández Alberto, Luengo Julián, Derrac Joaquín, García Salvador (2011) Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Multi Valued Log Soft Comput 17(2–3):255–287
3. Baskiotis N, 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, page 8. ACM, New York, NY, USA
4. Basu Mitra, Ho Tin Kam (2006) Data complexity in pattern recognition (advanced information and knowledge processing). Springe New York Inc., Secaucus, NJ
5. Baumgartner R, Somorjai RL (2006) Data complexity assessment in undersampled classification of high-dimensional biomedical data. Pattern Recognit Lett 12:1383–1389
6. Bensusan H, Kalousis A (2001) Estimating the predictive accuracy of a classifier. In EMCL '01: Proceedings of the 12th european conference on machine learning Springer, London, pp 25–36
7. Bernadó-Mansilla Ester, Ho Tin Kam (2005) Domain of competence of XCS classifier system in complexity measurement space. IEEE Trans Evol Comput 9(1):82–104
8. Brazdil P, Giraud-Carrier C, Soares C, Vilalta R (2009) Metalearning: applications to data mining. Cognitive Technologies, Springer
9. Cheeseman P, Kanefsky B, Taylor WM (1991) Where the really hard problems are. In: IJCAI'91: Proceedings of the 12th international joint conference on artificial intelligence. Morgan Kaufmann Publishers Inc, San Francisco, CA, pp 331–337
10. Demšar Janez (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
11. Derrac Joaquín, Triguero Isaac, García Salvador, Herrera Francisco (2012) Integrating instance selection, instance weighting, and feature weighting for nearest neighbor classifiers by coevolutionary algorithms. IEEE Trans Syst Man Cybern Part B 42(5):1383–1397
12. Dong M, Kothari R (2003) Feature subset selection using a new definition of classificabilty. Pattern Recognit Lett 24:1215–1225
13. Fernández A, García S, José M, del Jesús MJ, Francisco H (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
14. García Salvador, Cano José Ramón, Bernadó-Mansilla Esther, Herrera Francisco (2009) Diagnose of effective evolutionary prototype selection using an overlapping measure. Int J Pattern Recognit Artif Intell 23(8):2378–2398
15. García Salvador, Herrera Francisco (2008) An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J Mach Learn Res 9:2677–2694
16. Ho Tin Kam, Baird Henry S (1998) Pattern classification with compact distribution maps. Comput Vis Image Underst 70(1):101–110
17. Ho Tin Kam, Basu Mitra (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 24(3):289–300
18. Hoekstra A, Duin RPW (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, USA, pp 271–275
19. Kalousis A (2002) Algorithm selection via meta-learning. PhD thesis, Université de Geneve
20. Kuncheva LI, Rodrguez JJ (2013) A weighted voting framework for classifiers ensembles. Knowl Inf Syst (in press) doi:10.1007/s10115-012-0586-6
21. Lebourgeois F, Emptoz H (1996) Pretopological approach for supervised learning. In: ICPR '96: Proceedings of the international conference on pattern recognition (ICPR '96) volume IV-Volume 7472. IEEE Computer Society, Washington, DC, USA, pp 256–260
22. Lorena AC, Costa IG, Spolaôr N, de Souto MCP (2012) Analysis of complexity indices for classification problems: Cancer gene expression data. Neurocomputing 75(1):33–42
23. Lorena AC, de Carvalho ACPLF (2010) Building binary-tree-based multiclass classifiers using separability measures. Neurocomputing 73(16–18):2837–2845
24. Luengo Julián, Fernández Alberto, García Salvador, Herrera Francisco (2011) Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. Soft Comput 15(10):1909–1936
25. Luengo Julián, Herrera Francisco (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
26. Luengo Julián, Herrera Francisco (2012) Shared domains of competence of approximate learning models using measures of separability of classes. Inf Sci 185(1):43–65

27. Macia N, Bernadó-Mansilla E, Orriols-Puig A, Kam Ho T (2012) Learner excellence biased by data set selection: A case for data characterisation and artificial data sets. Pattern Recognit (in press). doi:10.1016/j.patcog.2012.09.022
28. McLachlan GJ (2004) Discriminant analysis and statistical pattern recognition. Wiley, New York
29. Mollineda RA, Sánchez JS, Sotoca JM (2005) Data characterization for effective prototype selection. In: Proceedings of the 2nd Iberian conference on pattern recognition and image analysis. Springer, pp 27–34
30. Okun Oleg, Priisalu Helen (2009) Dataset complexity in gene expression based cancer classification using ensembles of k-nearest neighbors. Artif Intell Med 45(2–3):151–162
31. Orriols-Puig Albert, Bernadó-Mansilla Ester (2008) Evolutionary rule-based systems for imbalanced data sets. Soft Comput 13(3):213–225
32. Orriols-Puig Albert, Casillas Jorge (2011) Fuzzy knowledge representation study for incremental learning in data streams and classification problems. Soft Comput 15(12):2389–2414
33. 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, CA, USA, pp 743–750
34. Platt J (1998) Machines using sequential minimal optimization. In: Schoelkopf B, Burges C, Smola A (eds) Advances in Kernel methods—support vector learning. MIT Press, Cambridge
35. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Mateo-California
36. Ramentol Enislay, Caballero Yaile, Bello Rafael, Herrera Francisco (2012) Smote-rsb *: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. Knowl Inf Syst 33(2):245–265
37. Sáez JA, Galar M, Luengo J, Herrera F (2013) Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. Knowl Inf Syst (in press) doi:10.1007/s10115-012-0570-1
38. Sáez José A, Luengo Julián, Herrera Francisco (2013) Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. Pattern Recognit 46(1):355–364
39. Sánchez José Salvador, Mollineda Ramón Alberto, Sotoca José Martínez (2007) An analysis of how training data complexity affects the nearest neighbor classifiers. Pattern Anal Appl 10(3):189–201
40. Singh S (2003) Multiresolution estimates of classification complexity. IEEE Trans Pattern Anal Mach Intell 25(12):1534–1539
41. Smith FW (1968) Pattern classifier design by linear programming. IEEE Trans Comput 17(4):367–372
42. Vainer Igor, Kaminka Gal A, Kraus Sarit, Slovin Hamutal (2011) Obtaining scalable and accurate classification in large scale spatio-temporal domains. Knowl Inf Syst 29(3):527–564
43. Vapnik VN (1998) Statistical learning theory. Wiley, New York
44. Wolpert David H (1996) The lack of a priori distinctions between learning algorithms. Neural Comput 8(7):1341–1390

Author Biographies

Julián Luengo received the M.S. degree in computer science and the Ph.D. from the University of Granada, Granada, Spain, in 2006 and 2011, respectively. He currently acts as an Assistant Professor in the Languages and Computer Systems area, a part of the Department of Civil Engineering at the University of Burgos, Spain. His research interests include machine learning and data mining, data preparation in knowledge discovery and data mining, missing values, noisy data, data complexity and fuzzy systems.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has been the supervisor of 28 Ph.D. students. He has published more than 250 papers in international journals. He received the following honors and awards: ECCAI Fellow 2009, IFSA Fellow 2013, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 Paper Award (bestowed in 2011) and 2011 Lotfi A. Zadeh Prize Best paper Award of the International Fuzzy Systems Association and 2013 AEPIA Award to a scientific career in Artificial Intelligence (September 2013). His current research interests include computing with words and decision making, bibliometrics, data mining, big data, cloud computing, data preparation, instance selection and generation, imperfect data, fuzzy rule-based systems, genetic fuzzy systems, imbalanced classification, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms, biometrics.