Information Sciences 185 (2012) 43–65

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Shared domains of competence of approximate learning models using measures of separability of classes ☆

Julián Luengo *, Francisco Herrera

Dept. of Computer Science and Artificial Intelligence, CITIC-UGR, 18071 Granada, Spain

ARTICLE INFO

Article history:
Received 7 November 2010
Received in revised form 7 June 2011
Accepted 18 September 2011
Available online 24 September 2011

Keywords:
Classification
Data complexity
Domains of competence
Multi Layer Perceptron
Radial Basis Function Network
Support Vector Machine

ABSTRACT

In this work we jointly analyze the performance of three classic Artificial Neural Network models and one Support Vector Machine with respect to a series of data complexity measures known as measures of separability of classes. In particular, we consider a Radial Basis Function Network, a Multi-Layer Perceptron, a Learning Vector Quantization, while the Sequential Minimal Optimization method is used to model the Support Vector Machine.

We consider five measures of separability of classes over a wide range of data sets built from real data which have proved to be very discriminative when analyzing the performance of classifiers. We find that two of them allow us to extract common behavior patterns for the four learning methods due to their related nature. We obtain rules using these two metrics that describe both good or bad performance of the Artificial Neural Networks and the Support Vector Machine.

With the obtained rules, we characterize the performance of the methods from the data set complexity metrics and therefore their common domains of competence are established. Using these domains of competence the shared good and bad capabilities of these four models can be used to know if the approximative models will perform well or poorly or if a more complex configuration of the model is needed for a given problem in advance.

1. Introduction

The use of Artificial Neural Networks (ANNs) in several tasks and fields is very common nowadays. Due their excellent adjusting capabilities, they have been successfully applied in the Data Mining ambit among many others [31], becoming a referent. In particular, ANNs have been succeeded in the classification task, within the Data Mining field. More recently, Support Vector Machines (SVMs) have become very popular in the same task due their high precision capabilities and their fast learning rate [11]. They share some common properties with respect to the ANNs, and it is usual to find them related in the specialized literature.

The research on ANNs in classification is very active nowadays as there are many recent contributions in this topic. Improvements in the weight adjusting for MLP in classification is a recurrent topic [1], even in challenging scenarios like imbalanced data [12,35]. RBFNs have been applied to incremental learning [30] and their adjustment is subject to recent enhancements [14]. Villmann et al. [37] exploit the synergy between fuzzy classification models and Learning Vector Quantization (LVQ) models. We can find from extensive comparisons of the LVQ models [17] to the study of the initial values in

☆ Supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01. J. Luengo holds a post-doctoral contract from University of Granada.
* Corresponding author.
E-mail addresses: julianlm@decsai.ugr.es (J. Luengo), herrera@decsai.ugr.es (F. Herrera).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.09.022

the MLP backpropagation scheme [40], including improvements on the RBFN learning methodology based on kernels [27]. Even the use of non-parametric statistics in the classification framework have been studied for MLPs and RBFNs models [22].

The use of SVMs is also very prolific in the same topic. Recent contributions include new developments in classification assuming fixed probability distributions of the data [38], the use of recursive SVMs to tackle the dimensionality problem [36], the study of formulations of the loss function in order to deal with imbalanced data sets directly [39] or simultaneously selects relevant features during classifier construction [24]. SVMs and ANNs are very related in their foundations and their integration has been already studied [10], and the analysis on the different properties they model and their advantages have been studied [15].

It is well-known that the prediction capabilities of ANNs and SVMs in classification are dependent on the problem's characteristics. Moreover determining the adequate topology or parameters of these models is usually determined by the intrinsic characteristic of the data set and it is difficult to estimate them a priori. An emergent field that uses a set of data complexity measures applied to the problem to try to capture different aspects or sources of complexity which are considered complicated to the classification task has recently arised [3]. Studies of data complexity metrics applied to particular classification's algorithms can be found [5,4,32,23].

In this work we are interested in determining the shared domains of competence of ANNs and SVMs using a particular set of metrics formerly known as *measures of separability of classes* [3]. We consider three classic models of ANNs, RBFN, MLP and LVQ; and the well-known Sequential Minimal Optimization (SMO) as a representative SVM model solving procedure. We want to describe and to analyze the known shared strengths and weakness of these four models by means of their domains of competence using the measures of separability of classes. They have proven to be very discriminative describing the domains of competence of other classification methods [5,23]. Once the shared domains of competence have been extracted, the data complexity measures contained in them provide a description of the problems' characteristics that are suitable or not for these four classifiers.

In order to perform this analysis, we create several binary classification data sets from real world problems, 438 in total, and compute the value of 5 measures of separability of classes proposed by Ho and Basu [18]. We analyze the intervals of measure values related to the created data sets, in which all the four methods perform well or bad, and then formulated a rule for such intervals. The obtained rules describe the ranges where some information and conclusions about the performance of ANN methods can be stated. With these rules the domains of competence of the learning method are defined following the approach presented in [23], that is, the intervals of the measures of separability of classes where the learning method performs well or badly. These domains of competence have a high support, supporting the assumption of the ANNs and the SVM being very close in nature, and giving an explanation in terms of characteristics of the data. The generalization abilities of these domains of competence are checked using a fresh bunch of 200 data sets.

The rest of this paper is organized as follows. In Section 2 we briefly describe the ANNs and the SVM models used. In Section 3 the considered complexity metrics are summarized as well as the state-of-the-art literature in the topic. Section 4 provides the motivation of this paper and the foundations of the domains of competence. In Section 5 we show the process used to build up the bunch of data sets used and the parameter adjusting process for the four models. Next, in Section 6 we introduce the motivation for the use of the measures of separability of classes and the analysis related to the extraction of the learning method's domains of competence. In Section 7 indications, criticisms and guidelines are provided. Finally, Section 8 concludes the paper.

2. Preliminaries: ANNs and SVM details

In this section, we will briefly describe the most important details of the particular implementations of ANN and SVM models that we have used. In Section 2.1 the Multi-Layer Perceptron model used is presented. Next in Section 2.2 the details of the RBFN model are described. Section 2.3 depicts the details of the LVQ model used. Finally, in Section 2.4 the SVM used is described. All the models described in this section are implemented in the KEEL software package.[1] Please refer to the associated references in order to obtain further details.

2.1. Multi-Layer Perceptron

A Multi-Layer Perceptron [25] is a classical model which consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in a layer has directed connections to the neurons of the subsequent layer. The units of these networks apply a sigmoid function as an activation function. Each connection of the units are related to a weight. Therefore we can define a weight vector as vector in the real euclidean space $\mathbb{R}^N$ where N is the number of weights and biases in the network.

$$\vec{w} = (\ldots, x_{ij}^l, w_{i+1j}^l, \ldots) \tag{1}$$

where w_{ij}^l is the weight from unit number i in layer number l to unit number j in layer number $l+1$.

[1] http://keel.es.

The neural network has attached a global error function $E(\vec{w})$ depending on all weights and biases, for example the standard least square function.

There are many algorithms for feedforward neural networks, like the standard back propagation algorithm. Most of the optimization methods used to minimize the error function are a local iterative process in which an approximation to the function in a neighborhood of the current point in the space is minimized, given by a first or second order Taylor expansion of the function. First an initial weight vector $\vec{w}_1$ is chosen. Then a search direction $\tilde{p}_k$ and a step size α_k are determined in step k so that $E(\vec{w}_k + \alpha_k\tilde{p}_k) < E(\vec{w}_k)$. If the final condition is not fulfilled then the weight vector is updated $\vec{w}_{k+1} = \vec{w}_k + \alpha_k\tilde{p}_k$ and a new step $k+1$ begins.

However, more recent and scalable approaches can be used and with better convergence. In the case of the back propagation algorithm, the seach direction $\tilde{p}_k$ can be set to the negative gradient $-E'(\vec{w})$ and the step size α_k to a constant ε, considering the linear approximation $E(\vec{w}+\vec{y}) \approx E(\vec{w}) + E(\vec{w})^T y$. An attempt to improve the convergence rate for back propagation is the addition of the momentum term, but also increases the user-dependant parameters.

The scaled conjugate gradient (SCG) employs the above general optimization strategy but choose the search direction more carefully by using information from the second order approximation

$$E(\vec{w}+\vec{y}) \approx E(\vec{w}) + E(\vec{w})^T\vec{y} + \frac{1}{2}y^T E''(\vec{w}\vec{y}). \tag{2}$$

SCG does not use the line search per learning iteration in order to estimate the step size, but a different approach which decreases the number of user-dependant parameters needed. The step size $\tilde{s}_k = E''(\vec{w}_k)\tilde{p}_k$ is estimated as

$$\tilde{s}_k = E''(\vec{w}_k)\tilde{p}_k \approx \frac{E'(\vec{w}_k + \sigma_k\tilde{p}) - E'(\vec{w}_k)}{\sigma_k} + \lambda_k\tilde{p}_k \quad 0 < \sigma_k \ll 1 \tag{3}$$

which tends in the limit to the true value of $E''(\vec{w}_k)\tilde{p}_k$, and λ_k is adjusted in each iteration depending on the sign of $\delta_k = \tilde{p}_k^T\tilde{s}_k$. If $\lambda_k \leqslant 0$ then the Hessian matrix $E''(\vec{w})$ is not positive definite and λ_k is raised and $\tilde{s}_k$ estimated again. The λ_k value scales the step size in each iteration, giving the name to the algorithm.

2.2. Radial Basis Function Network

Radial Basis Function Networks [8,9] are well suited for function approximation and pattern recognition due to their simple topological structure and their ability to reveal how learning proceeds in an explicit manner. A RBF is a function which has been built into a distance criterion with respect to a centre. Different basis functions like thin-plate spline functions, multiquadratic functions, inverse multiquadratic functions and Gaussian functions have been proposed for the hidden-layer neurons, but normally the selected one is the Gaussian function. Compared with other types of Artificial Neural Networks, such as feed-forward networks, the RBFN requires less computation time for learning and also has a more compact topology. RBFs have been applied in the area of ANNs where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in Multi-Layer Perceptrons. The original RBF method has been traditionally used for strict multivariate function interpolation [29] and for this fact, it requires as many RBF neurons as data points. Broomhead and Lowe removed this strict interpolation restriction [8] and provided a neural network architecture where the number of RBF neurons can be far less than the data points. A RBFN mainly consists of two layers, one hidden-layer and one output layer.

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

$$h_i(x) = exp\left[\frac{(x-v_i)^2}{2\sigma_i^2}\right] \tag{4}$$

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

$$z(x) = \frac{\sum_i^k h_i(x)w_i}{\sum_i^k h_i(x)} \tag{5}$$

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

$$\sigma_i = \sum_{j=1}^{N}\frac{d(v_i, v_j)}{N} \tag{6}$$

The network is initialized with a K-means clustering algorithm, setting the number of clusters equal to the number of neurons. The initial centroids are set to random chosen examples, which are all different. By successively iterations, the neurons are adjusted using the Euclidean distance till the centroids (i.e. the neurons) do not change.

Once the centers and radius of the neurons has been initialized, the output's weight matrix can be optimized by means of supervised training. For each train example x_i and expected output t_i, we compute the output of the hidden layer's neurons, the vector h. Next, we compute the output of the network y and compare it with the expected output t, and adjust each

weight in w to reduce the Mean Square Error (MSE) with the Least Mean Squares algorithm (LMS). This method implies the use of gradient descent (delta rule) and adjusting the weights:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(t_j - y_j)h_i \tag{7}$$

where η is the learning rate ($\eta \ll 1.0$). This process is repeated for each train example, until the max iteration limit is reached. The center and radius of the neurons is adjusted as well to minimize the output error. We use the error derivative with respect to these parameters, in a fashion similar to that of backpropagation. With $i = 1, \ldots, m$ hidden neurons, $j = 1, \ldots, p$ inputs and $k = 1$ output, we update both parameters simultaneously from iteration n to $n + 1$ with:

$$v_{ij}(n+1) = v_ij(n) + \eta_c \frac{\left[\sum_k (t_k - y_k)w_{ik}\right] h_i (x_j - v_{ij}(n))}{(\sigma_i(n))^2} \tag{8}$$

$$\sigma_i(n+1) = \sigma_i(n) + \eta_\sigma \frac{\left[\sum_k (t_k - y_k)w_{ik}\right] h_i \|x - v_i(n)\|^2}{(\sigma_i(n))^3} \tag{9}$$

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

2.3. Learning Vector Quantization

The Learning Vector Quantization (LVQ) [6] family comprises a large spectrum of competitive learning schemes. One of the basic designs that can be used for classification is the LVQ1 algorithm.

An initial set of labeled prototypes is picked by first specifying $n_p \geqslant c$. Then n_p elements are randomly selected from X to be the initial prototypes, so each class is represented by at least one prototype. LVQ1 has three additional parameters specified by the user: the learning rate $\alpha_k \in (0,1)$, a constant $\eta \in (0,1)$, and the terminal number of iterations T. The standard competitive learning update equation is then used to alter the prototype set. If the closed prototype for input x is the vector $v_{i,old}$, the LVQ1 update strategy is

$$v_{i,new} = v_{i,old} + \alpha_k(x - v_{i,old}) \quad \text{when} \quad \ell(v_{i,old}) = \ell(x) \tag{10}$$

or

$$v_{i,new} = v_{i,old} - \alpha_k(x - v_{i,old}) \quad \text{when} \quad \ell(v_{i,old}) \neq \ell(x) \tag{11}$$

Eq. (10) rewards the winning prototype for getting the correct label by letting it migrate towards the input vector, while Eq. (11) punishes the winning prototype for not labeling the current input correctly by repelling it away from the input vector. In our experiments, the learning rate was updated after each presentation of a new vector using the formula $\alpha_{k+1} = \eta\alpha_k$, $k = 1, \ldots, n-1$; and was restored to the initial, user specified value of α_1 at the end of each pass through X. Before each pass through LVQ1, X is randomly permuted to avoid dependence of the extracted prototypes on the order of inputs. LVQ1 terminates when either (i) there is no misclassifications in a whole pass through X (and hence, the extracted prototypes are a consistent reference set); or (ii) the prespecified terminal number of iterations is reached.

2.4. Support Vector Machine

A Support Vector Machine (SVM) [11] constructs a hyperplane or set of hyperplanes in a high-dimensional space. A good separation is achieved by the hyperplane that has the largest distance to the nearest training datapoints of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. In order to solve the quadratic programming (QP) problem that arises from SVMs, there are many techniques mostly reliant on heuristics for breaking the problem down into smaller, more-manageable chunks.

The QP problem to train an SVM is shown below:

$$\begin{aligned} &max_\alpha W(\alpha) = \sum_{i=1}^{l} \alpha_i - \tfrac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j k(x_i, x_j)\alpha_i\alpha_j \\ &\quad 0 \leqslant \alpha_i \leqslant C, \quad \forall i \\ &\quad \sum_{i=1}^{l} y_i\alpha_i = 0 \end{aligned} \tag{12}$$

where $x_i, x_j \in \mathbb{R}^n$.

A point is an optimal point of (12) if and only if the Karush–Kuhn–Tucker (KKT) conditions are fulfilled and $Q_{ij} = y_i y_j k(x_i x_j)$ is positive semi-definite. The KKT conditions are simple; the QP problem is solved when, for all i:

$$\begin{aligned} &\alpha_i = 0 \Rightarrow y_i f(x_i) \geqslant 1 \\ &0 < \alpha_i < C \Rightarrow y_i f(x_i) = 1 \\ &\alpha_i = C \Rightarrow y_i f(x_i) \leqslant 1 \end{aligned} \tag{13}$$

There are several proposals in the literature used to solve this problem. Initial ones like the projected conjugate gradient (PCG) use a "chunking" algorithm based on the fact that the value of the quadratic form is the same if you remove the rows and columns of the matrix that correspond to zero Lagrange multipliers. Therefore, the large QP problem can be broken down into a series of smaller QP problems, whose ultimate goal is to identify all of the non-zero Lagrange multipliers and discard all the zero Lagrange multipliers. At every step, chunking procedures solves a QP problem that consists of the every non-zero Lagrange multiplier from the last step, and the M worst examples that violate the KKT conditions (13). However, modern procedures suggest a new strategy, avoiding the "chunking" process and speeding up the whole process. A common method for solving the QP problem is the Platt's Sequential Minimal Optimization algorithm considered in this paper [28]. It breaks the problem down into 2-dimensional sub-problems that may be solved analytically, eliminating the need for a numerical optimization algorithm as done in PCG.

In order to solve for the two Lagrange multipliers α_1 and α_2, SMO first computes the constraints on these multipliers and then solves for the constrained maximum. The bound constraints in Eq. (12) cause the Lagrange multipliers to lie within a box, while the linear equality constraint in Eq. (12) causes the Lagrange multipliers to lie on a diagonal line.

The ends of the diagonal line segment can be expressed quite simply. Without loss of generality, the algorithm first computes the second Lagrange multiplier α_2 and computes the ends of the diagonal line segment in terms of α_2. If the target y_1 does not equal the target y_2, then the following bounds apply to α_2:

$$L = \max(0, \alpha_2^{old} - \alpha_1^{old}), \quad H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}) \tag{14}$$

If the target y_1 equals the target y_2, then the following bounds apply to α_2:

$$L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C), \quad H = \min(C, \alpha_2^{old} + \alpha_1^{old}) \tag{15}$$

The second derivative of the objective function along the diagonal line can be expressed as:

$$\eta = 2k(x_1, x_2) - k(x_1, x_1) - k(x_2, x_2) \tag{16}$$

The next step of SMO is to compute the location of the constrained maximum of the objective function in Eq. (12) while allowing only two Lagrange multipliers to change. Under normal circumstances, there will be a maximum along the direction of the linear equality constraint, and η will be less than zero. In this case, SMO computes the maximum along the direction of the constraint.

Therefore SMO will always optimize two Lagrange multipliers at every step, with one of the Lagrange multipliers having previously violated the KKT conditions before the step. That is, SMO will always alter two Lagrange multipliers to move uphill in the objective function projected into the one-dimensional feasible subspace. SMO will also always maintain a feasible Lagrange multiplier vector. Therefore, the overall objective function will increase at every step and the algorithm will converge asymptotically.

3. Data complexity: measures of separability of classes

In this section we first present a short review on recent studies on data complexity in Section 3.1, then we describe the data complexity measures that we have used in this work in Section 3.2.

3.1. Recent studies in data complexity

As we have mentioned, data complexity measures are a series of metrics that quantify characteristics which imply some difficulty to the classification task. In the following we gather several recent publications related to these complexity measures and their applications. They can show a picture of the most recent developments in the topic.

- Ho and Basu propose some complexity measures for binary classification problems [18], gathering metrics of three types: overlaps in feature values from different classes; separability of classes; and measures of geometry, topology, and density of manifolds.
- Singh offers a review of data complexity measures and proposes two new ones [33].
- Bernadó and Ho investigate the domain of competence of XCS by means of a methodology that characterizes the complexity of a classification problem by a set of geometrical descriptors [5].
- Li et al. [21] analyze some omnivariate decision trees using the measure of complexity based on data density proposed by Ho and Basu.
- Baumgartner and Somorjai define specific measures for regularized linear classifiers [4], using Ho and Basu's measures as reference.
- Sánchez et al. analyze the effect of data complexity on the nearest neighbors classifier [32].
- Dong and Kothari propose [13] a feature selection algorithm based on a complexity measure defined by Ho and Basu.
- Mollineda et al. [26] extend some of Ho and Basu's measure definitions for problems with more than two classes. They analyze these generalized measures in two classic Prototype Selection algorithms and remark that Fisher's discriminant ratio is the most effective for Prototype Selection.

- Kim and Oommen [20] analyze how to use prototype selection in order to decrease the computation time of several data complexity measures, without severely affecting the outcome with respect to the complete data set.
- García et al. [16] analyze the behavior of the evolutionary prototype selection strategy, considering a complexity measure for classification problems based on overlapping.
- Luengo and Herrera [23] characterized the performance of the FH-GBML Fuzzy Rule Based Classification System by means of ad hoc intervals of the data complexity measures [18].

3.2. Measures of separability of classes

For our study, we will consider five measures of separability of classes, from the total twelve as described by Ho and Basu [18]. These measures have proven to be the most discriminative in the classification task [5,23] due to their high relationship with the class separability and overlapping characteristics. Please note that most of these data complexity measures are only well defined for binary problems.

The measures of separability of classes provide indirect characterizations of class separability. They assume that a class is made up of single or multiple manifolds that form the support of the probability distribution of the given class. The shape, position and interconnectedness of these manifolds give hints on how well two classes are separated, but they do not describe separability by design. The particular descriptions of the measures follows:

L1: *minimized sum of error distance by linear programming*. Linear classifiers can be obtained by a linear programming formulation proposed by Smith [34]. The method minimizes the sum of distances of error points to the separating hyperplane (subtracting a constant margin):

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

where $\mathbf{a}$, $\mathbf{b}$ are arbitrary constant vectors (both chosen to be 1), $\mathbf{w}$ is the weight vector to be determined, $\mathbf{t}$ is an error vector, and $\mathbf{Z}$ is a matrix where each column $\mathbf{z}$ is defined on an input vector $\mathbf{x}$ (augmented by adding one dimension with a constant value 1) and its class C (with value C_1 or C_2) as follows:

$$\begin{cases} \mathbf{z} = +\mathbf{x} & \textbf{if } \mathbf{C} = \mathbf{C_1} \\ \mathbf{z} = -\mathbf{x} & \textbf{if } \mathbf{C} = \mathbf{C_2} \end{cases}$$

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 note that this measure can be heavily affected by the presence of outliers in the data set.

L2: *error rate of linear classifier by Linear Programming* (LP). This measure is the error rate of the linear classifier defined for L1, measured with the training set. With a small training set this can be a severe underestimate of the true error rate.

N1: *fraction of points on class boundary*. This method constructs a class-blind minimum spanning tree over the entire data set, and counts the number of points incident to an edge going across the two classes. The fraction of such points over all points in the data set is used as a measure.

N2: *ratio of average intra/inter class nearest neighbor* (NN) *distance*. For each input instance x_p, we calculate the distance to its nearest neighbor within the class (intraDist (x_p)) and the distance to nearest neighbor of any other class (interDist (x_p)). Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example, i.e.,

$$N2 = \frac{\sum_{i=0}^{m} intraDist(x_i)}{\sum_{i=0}^{m} interDist(x_i)}$$

where m is the number of examples in the data set. This metric compares the within-class spread with the distances to the nearest neighbors of other classes. Low values of this metric suggest that the examples of the same class lie closely in the feature space. Large values indicate that the examples of the same class are disperse.

N3: *error rate of 1-NN classifier*. This is simply the error rate of a nearest-neighbor classifier measured with the training set. The error rate is estimated by the leave-one-out method. The measure denotes how close the examples of different classes are. Low values of this metric indicate that there is a large gap in the class boundary.

4. Domains of competence of classifiers

In this section the motivation behind the proposal is presented in Section 4.1. The concept of the domains of competence of the learning methods used in this paper is given in Section 4.2.

4.1. Motivation

One first indicator of the performance of the ANN or SVM considered could be the average accuracy obtained in the training partitions. However, it is well known that it is possible to increase such accuracy in detriment of the test accuracy. This phenomena is usually known as over-fitting in the classification topic, and it discourages the use of the training accuracy as a precise indicator of the true performance of the model.

Figs. 1–4 contains the results of the four models showing the training and test accuracy over all the initial 438 data sets described in Section 5.1, plotted in ascending training accuracy value for the RBFN, MLP, LVQ and SVM models respectively.

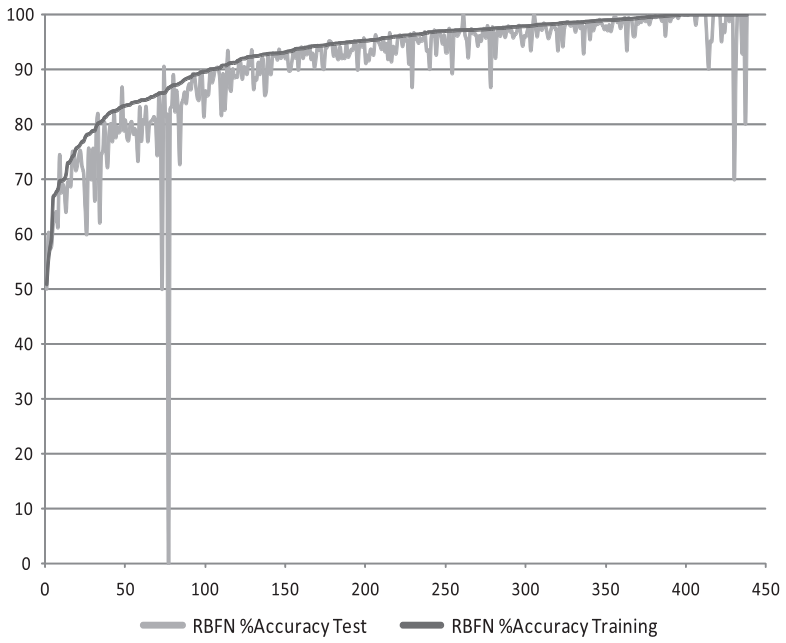

Fig. 1. Accuracy in training/test for RBFN sorted by training accuracy.

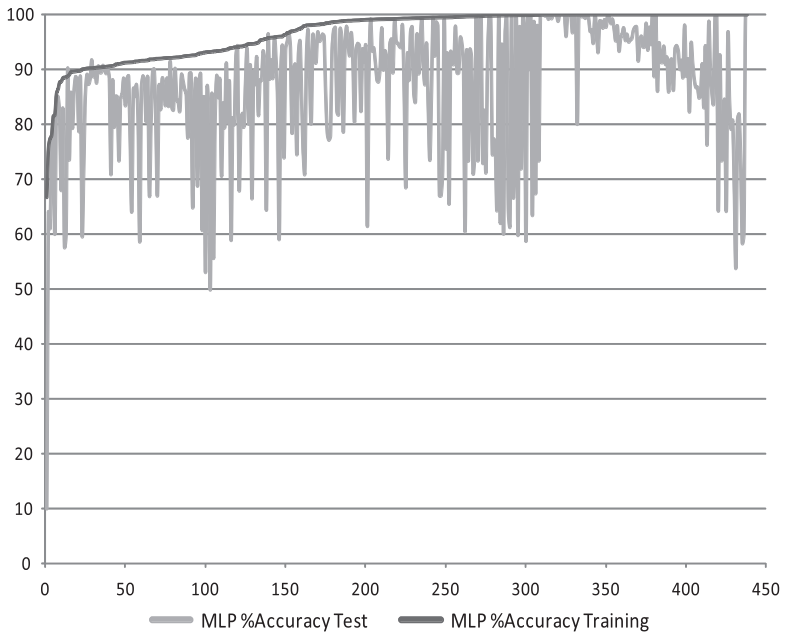

Fig. 2. Accuracy in training/test for MLP sorted by training accuracy.

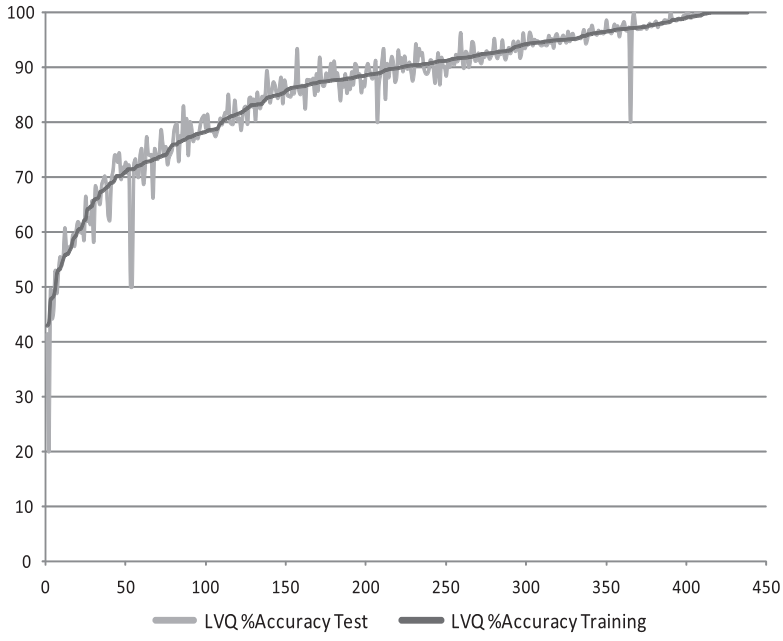

Fig. 3. Accuracy in training/test for LVQ sorted by training accuracy.

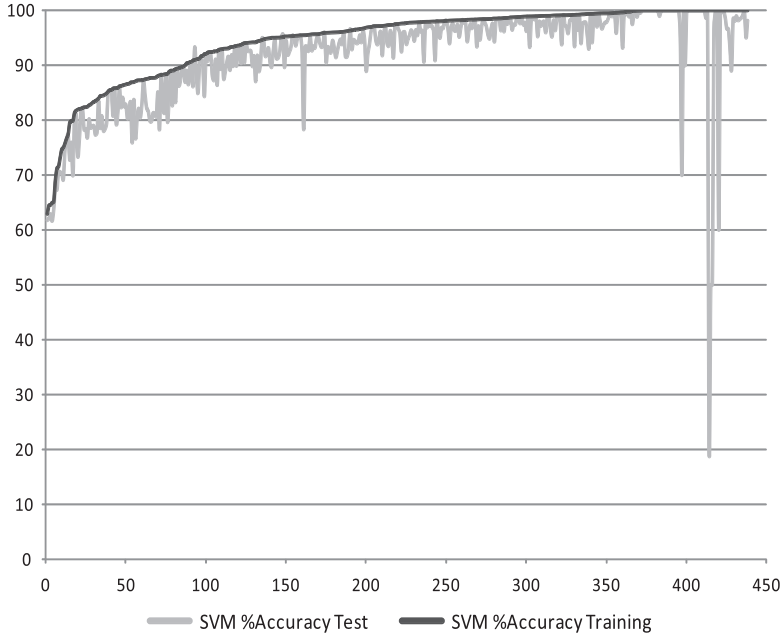

Fig. 4. Accuracy in training/test for SVM sorted by training accuracy.

The dark line in these figures represent the training values, and it is possible to notice the variability of test values associated. We want to point out that we can find over-fitting in all four Figs. 1–4 especially in the highest training values.

An alternative option is to use the data complexity measures to analyze when the model produces a good or bad outcome for the data set. These metrics measure aspects on the class separability of the data set that do not depend on the classification method used afterwards but on the characteristics of the data set itself. The good or bad performance of a learning method can be related to regions of the measures of separability of classes using some representative data sets. Then these regions can be used to predict if the classification method is suitable or not for other data sets presenting values in these regions without running the classification method over them as developed in [23].

At this point the necessity of the joint analysis of the domains of competence of two different classifiers arises. If two classification methods are similar in their operation, it can be expected that the good and bad performance regions for both methods will present a large overlapping, stating their related nature. As the data complexity measures analyze aspects of the data, it is possible to use the information provided by the data complexity measures based on geometrical characteristics of the data in order to give a description of the shared performance of the classifiers. This will be developed in this paper by means of the shared regions of the domains of competence of the four approximate learning methods considered: RBFN, MLP, LVQ and SVM.

4.2. 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. These problems share some aspects that explain the good or bad performance of the method on them. There are some previous approaches in the literature which define the domains of competence of a learning method based on the data complexity measures. As mentioned, this may be a key question when training a classification method takes a long time, due to the complexity of the learning algorithm or the size of the data. Problems denoted as difficult *a priori* may also require a different parameter configuration in respect to simpler ones. The data complexity measures also give hints on the nature of the data, and specific efforts can be made in order to improve the method's performance in the difficult regions.

Bernadó-Mansilla and Ho [5] initially defined the concept of domains of competence for the XCS classifier. Their domains of competence indicate the "region" of the complexity space of problems adequate to the learning method characteristics. Such a characterization could be useful for focusing the efforts of the learning algorithm's improvements on those difficult areas. In order to establish such limits, they used six of the twelve data complexity measures presented in [18], and related the error rate of XCS to high or low values of the data complexity metrics. They also observed which kind of metrics best discriminate between difficult and easy problems for XCS.

In [23] their notion of domains of competence for the FH-GBML method is extended eliminating the constraints on the low or high values. Specific intervals of the measures in which the FH-GBML method performs well or poorly (instead of relating its performance to "high" or "low" values) are extracted ad hoc. These intervals were coded as rules which constitute the domains of competence of the learning method.

Please note that the domains of competence of two different classifiers cannot be compared in order to determine the best classifier. Only their shared strengths and weakness can be pointed out. The choice between classifiers based on their accuracy is directly related to the Meta Learning approach [7]. Many problems arise when trying to determine the best algorithm [19] and no satisfactory solution has been proposed yet.

For these reasons the predictive model selection is out of the scope of this paper. In this paper we aim to relate the domains of competence obtained for individual classifiers and to analyze the shared regions of these domains of competence. These shared regions can point out the problems that are well suited for all the considered learning methods and help to identify why they are related.

5. Experimental framework

In order to obtain the characterization of the performance of the ANNs models and the SVM in subsequent sections, their parameters and the problems used as test-bed must be stated. Therefore in this section we first provide details of the data sets used for the experimentation in Section 5.1. Next the parameter configuration for the methods studied is presented in Section 5.2.

5.1. Data sets generation

We first evaluate the ANNs and the SVM over a set of 438 binary classification problems, in order to obtain their domains of competence. A big quantity of data sets is required because we need to obtain a rich variety of values for each measure of separability of classes. It is not possible to generate data sets with specific values of data complexity measures at this point, so we create many data sets from different original problems in order to fill the value range of each measure.

These first 438 problems are generated from pairwise combinations of the classes of 21 problems from the University of California, Irvine (UCI) repository [2]. These are *iris, wine, new-thyroid, solar-flare, led7digit, zoo, yeast, tae, balanced, car, contraceptive, ecoli, hayes-roth, shuttle, australian, pima, monks, bupa, glass, haberman* and *vehicle*. In order to do that, we construct several new data sets with the combination of the examples from two different classes. This will result in a new data set with only 2 classes and with the original examples which had two such classes as output. We perform this process for every possible pairwise combination of classes, and we compute the five measures of separability of classes for every data set. If an obtained data set with this procedure proves to be linearly-separable, we discard it. The complexity measure L1 indicates if a problem is linearly-separable when its value is equal to zero.

This method for generating binary data sets is limited by the proper combinatorics, and we can only obtain approximately 200 new data sets with the original 21 data sets with this first approach. In order to obtain more data sets, we group the

Table 1
Parameters used for the models.

MLP	Hidden layers = 1, Neurons per layer = 40
RBFN	Neurons = 50
LVQ	$n_p = 40$, $\alpha = 0.3$, $\eta = 0.8$
SVM	$C = 1$, tolerance = 0.001, kernel = Puk, $\epsilon = 1^{-12}$, $\omega = 1$, $\sigma = 1$

Table 2
Global average training and test accuracy/std. dev. for RBFN, MLP, LVQ and SVM.

	Global % Accuracy training global training std. dev.	Global % accuracy test global test std. dev.
RBFN	93.12%	90.65%
	7.99	10.42
MLP	96.82%	87.02%
	4.40	11.68
LVQ	86.44%	86.35%
	11.93	12.46
SVM	94.72%	91.99%
	6.07	9.13

classes two by two, that is, we create a new binary data set, and each of its two classes are the combination of two original classes each. For this second approach we have used *ecoli*, *glass* and *flare* data sets, since they have a high number of class labels, obtaining 238 new ones which are added to the initial 200 ones to sum up the aforementioned 438 data sets. Again, those generated data sets with a L1 value of zero were discarded.

A second bunch of data sets will be considered in order to validate the results obtained in our analysis apart from the 438 initial ones. We have applied the same methodology of grouping the classes two by two to the *yeast, flare* and *car* data sets. With these last three data sets we have obtained another 200 non-linearly separable data sets used for validating the rules obtained in our analysis, for a total of 638 data sets.

In order to measure the ANNs and the SVM accuracy performance in each data set, we have applied a ten-fold cros validation scheme.

5.2. Parameter settings

The parameter selection for ANN and SVM models is a key question in order to obtain a good performance and avoid overfitting. The common process implies to train the model with different topologies and learning parameters over the problems (i.e. using a parameter grid) and to choose the best configuration.

When analyzing the performance of these models with respect to the measures of separability of classes we consider to use the same configuration for all the data sets in order to relate results across all the data sets. Different parameter configurations of the models may produce very different outcomes, and we want to relate the good or bad outputs to the measures of separability of classes values as independently of external factors as possible. Moreover, a particular configuration of the models for each data set built as indicated in the previous section would be very time consuming.

Therefore, we have selected the same parameters for all the 438 initial data sets in each model, trying to obtain a good overall performance. These parameters have been chosen from a range of possibilities, finally using those which offer the best average performance all over the initial 438 data sets. These ranges follows:

- MLP, RBFN and LVQ number of neurons: from 10 to 50, using increments of 10.
- SVM C parameter: from 1 to 100, using increments of 10.
- MLP, RBFN, LVQ and SVM learning parameters (α, η, ω, σ): from 0.1 to 1, using increments of 0.1.

The final best overall parameter configuration as required by KEEL software for each model is summarized in Table 1.

In Table 2 we have summarized the global training and test accuracy for the initial 438 data sets obtained by the ANN methods and the SVM with these parameter configuration.

6. Analysis of the learning methods with measures of separability of classes

In this study, our aim is to analyze the characterization of the performance of the four learning methods by means of the data complexity measures. In order to do this analysis, we divide this section into the following subsections. In Section 6.1 we describe the process used in order to obtain the discriminative measures of separability of classes. Next we determine several rules based on intervals of the selected data complexity measures for the ANNs and SVM in Section 6.2. Then we

determine the shared domains of competence of the four models from the single rules in Section 6.3 and validate them in Section 6.4.

6.1. Selection of the discriminative measures of separability of classes

Given the initial 438 data sets described in Section 5.1, we will obtain 438 values of each measure of separability of classes. We try to extract regions of these measures in which all the four models behave well or badly. In order to do so, for each complexity measure the data-sets are sorted by the ascending value of such complexity measure, and put altogether in a figure. In the *X* axis the data sets are depicted with the same distance between them. The reason to do so is to avoid huge gaps between metric values to distort the interval extraction procedure described next. The *Y* axis depicts the accuracy obtained both in training and test for the model.

Once the data set have been arranged as described above, we have extracted different *ad hoc* intervals which present *good* or *bad performance* for the ANNs and the SVM *simultaneously*. The considered criteria for this analysis follows as a representative example of good and bad performance in classification. Nevertheless it must be stated that it can be adjusted to each one's necessities.

- We understand for *good performance* an average high test accuracy in the interval, at least 90%.
- By *bad performance* we refer to the presence of over-fitting and/or average low test accuracy in the interval (under 80%).

These intervals must also contain a significative number of data sets in order to avoid biases introduced by small good/bad intervals located in bad/good regions respectively. At least the 15% of the total number of data sets is required to consider an interval big enough to be extracted. On the other hand, if only a few data sets do not verify these indications, the interval can be extracted as well, as these data sets would act as outliers in this case.

It is not always is possible to extract shared intervals for all the four models with the mentioned characteristics. For example, in Fig. 5 we show an example of a figure without any quality interval for MLP. Fig. 6 depicts the results for SVM sorted by the same L1 measure. It can be seen that while SVM shows good performance for low L1 values with respect to higher values, this distinction cannot be made for MLP. Therefore this measure cannot be considered for extracting shared intervals between these two methods (thus being discarded for RBFN and LVQ as well).

For some data complexity measures there are one or more intervals of either good or bad performance for all the four classifiers. Table 3 summarizes which measures are discriminative to distinguish between good and bad classifier's performance for every approximate learning method and which ones are not. Thus only for the N1 and N3 measures of separability of classes we can extract common intervals of good and bad performance for the four classifiers. These two measures are specially designed to estimate both the proximity of examples of different classes in their boundaries and to indicate wide gaps between such boundaries.

In Figs. 7–14 we present the figures for the N1 and N3 measures in which we have extracted the *ad hoc* intervals which present good or bad performance for all the ANNs and SVM with the same size, using a vertical line to delimit them.

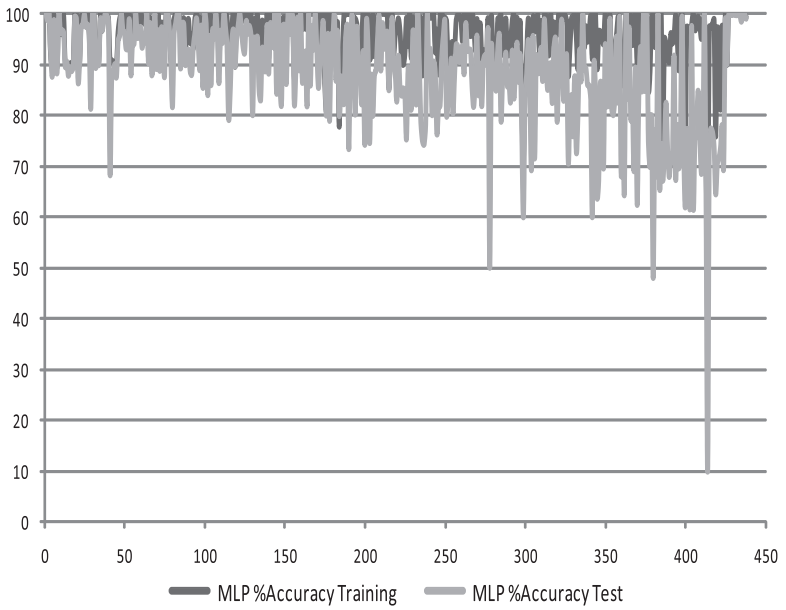

Fig. 5. MLP accuracy in training/test sorted by L1.

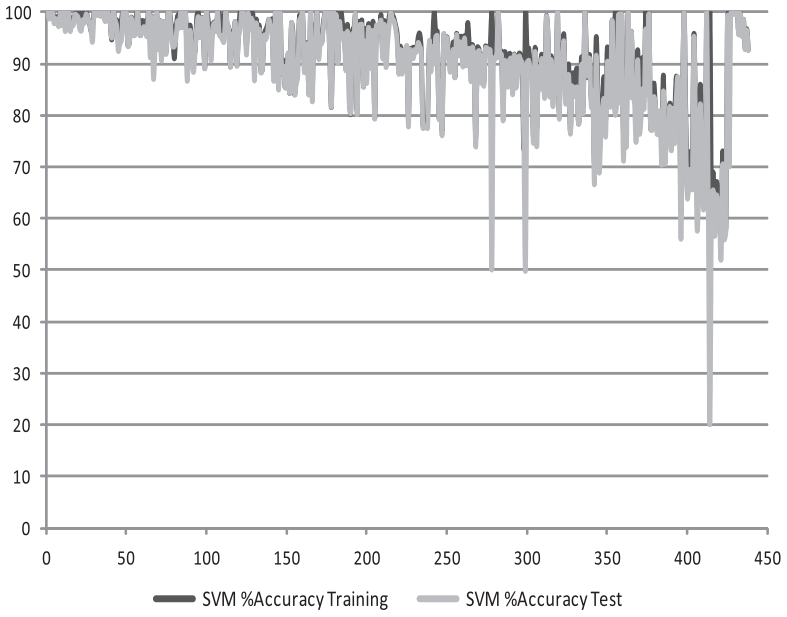

Fig. 6. SVM accuracy in training/test sorted by L1.

Table 3
Measures of separability of classes which show differentiated performance for the four classifiers (GP stands for "good performance", "BP stands for "bad performance").

	RBFN	MLP	LVQ	SVM
L1	GP at low values BP at high values	Not discriminative	Not discriminative	GP at low values BP at high values
L2	GP at low values BP at high values	Not discriminative	BP at high values	GP at low values BP at high values
N1	GP at low values BP at high values	GP at low values BP at high values	GP at low values BP at high values	GP at low values BP at high values
N2	GP at low values BP at high values	GP at low values BP at high values	Not discriminative	GP at low values BP at high values
N3	GP at low values BP at high values	GP at low values BP at high values	GP at low values BP at high values	GP at low values BP at high values

6.2. Modeling the algorithm's performance

In Table 4 we summarize the particular boundary values of the ad hoc intervals depicted in Figs. 7–14 for the N1 and N3 complexity measures.

From these ad hoc intervals it is possible to construct several rules that model the performance of the ANNs and the SVM. In Table 5 we have summarized the rules derived from Table 4 using the intervals as the antecedents of the rules. Given a particular data set X, we get the complexity measure of X with the notation $CM[X]$. Table 5 is organized with the following columns.

- The first column corresponds to the identifier of the rule for further references.
- The "Rule'" column presents the rule itself.
- The third column "Support" presents the percentage of data sets which verifies the antecedent of the rule.
- The column "Model" identifies the classification model to which this row refers to.
- The column "% Training" shows the average training accuracy of the classifier for the data sets that fall in the interval.
- The column "Training Diff." contains the difference between the value of the column "% Training" and the global training accuracy presented in Table 2 for the classifier.
- The column "% Test" shows the average test accuracy of the classifier for the data sets that fall in the interval.
- The column "Test Diff." contains the difference between the value of the column "% Test" and the global test accuracy presented in Table 2 for the classifier.

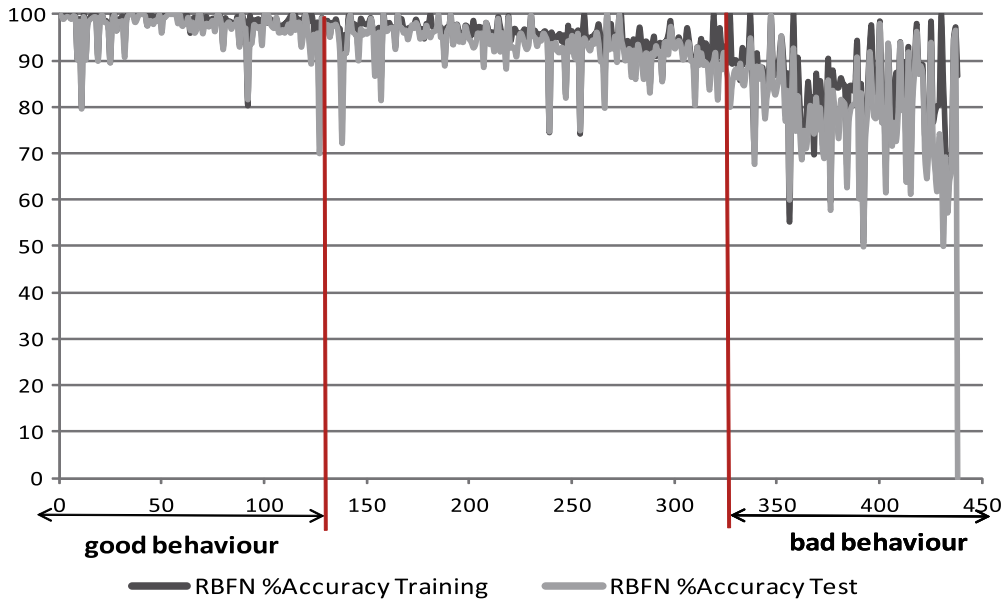

Fig. 7. RBFN accuracy in training/test sorted by N1.

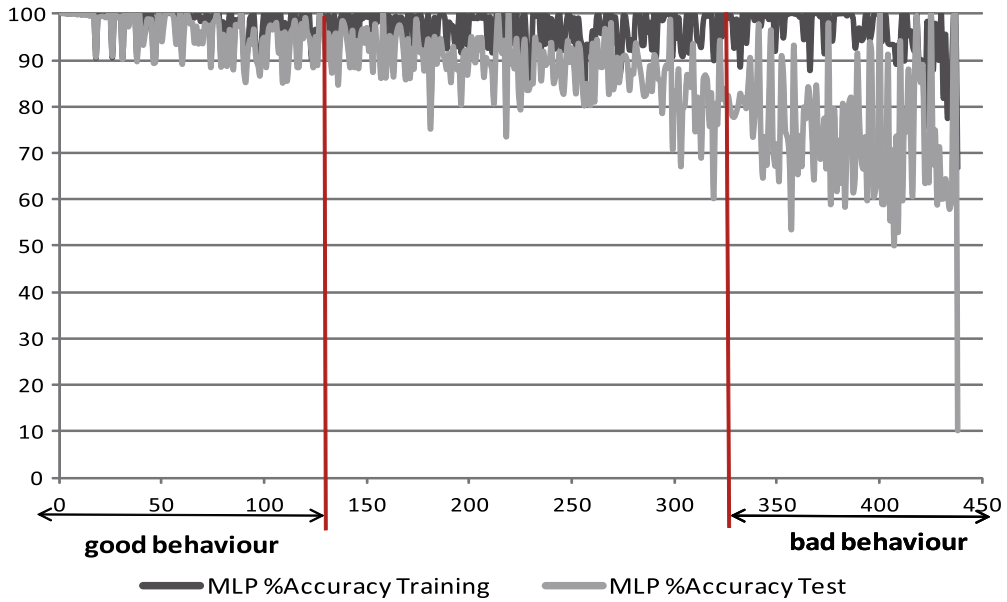

Fig. 8. MLP accuracy in training/test sorted by N1.

The positive rules (denoted with a "+" symbol in their identifier) always show a positive difference with the global average, both in training and test accuracy. The negative ones (with a "−" symbol in their identifier) verify the opposite case. The support of the rules shows us that we can characterize a wide range of data sets and obtain significant differences in accuracy.

From this set of rules we can state the following:

- A low value of N1 results in a good performance of all the learning models considered. That means that those problems with a low percentage of instances of different classes that are closer than to examples of their own class are adequate for the ANNs and SVM. They are characterized by a good separation of examples in the class boundaries.
- A low value of N3 results in a good performance of all the learning models considered. When a wide gap between the class boundary is present, it is easy to shape the separation between the classes and to obtain a good model for the problem.

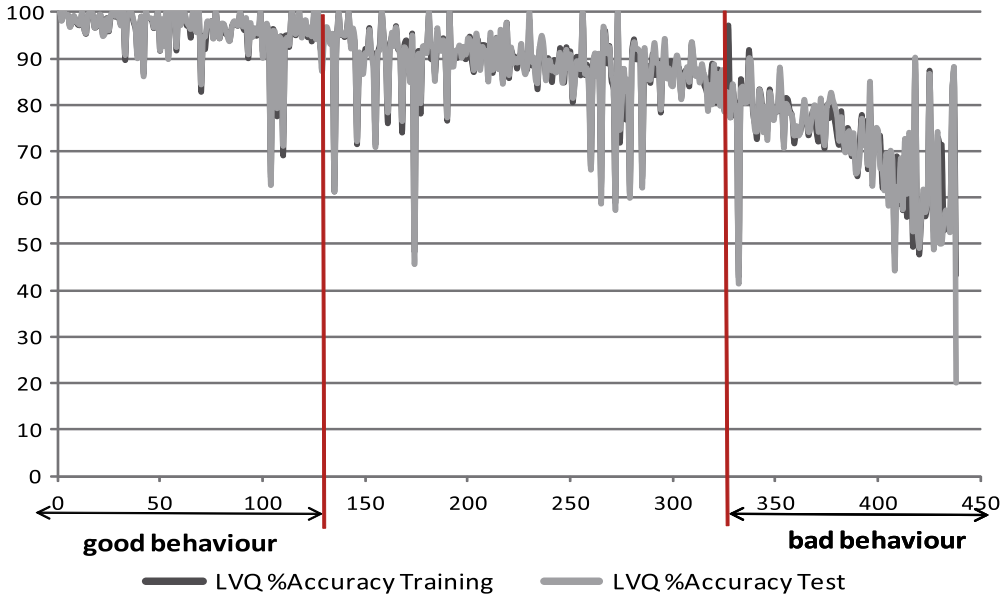

Fig. 9. LVQ accuracy in training/test sorted by N1.

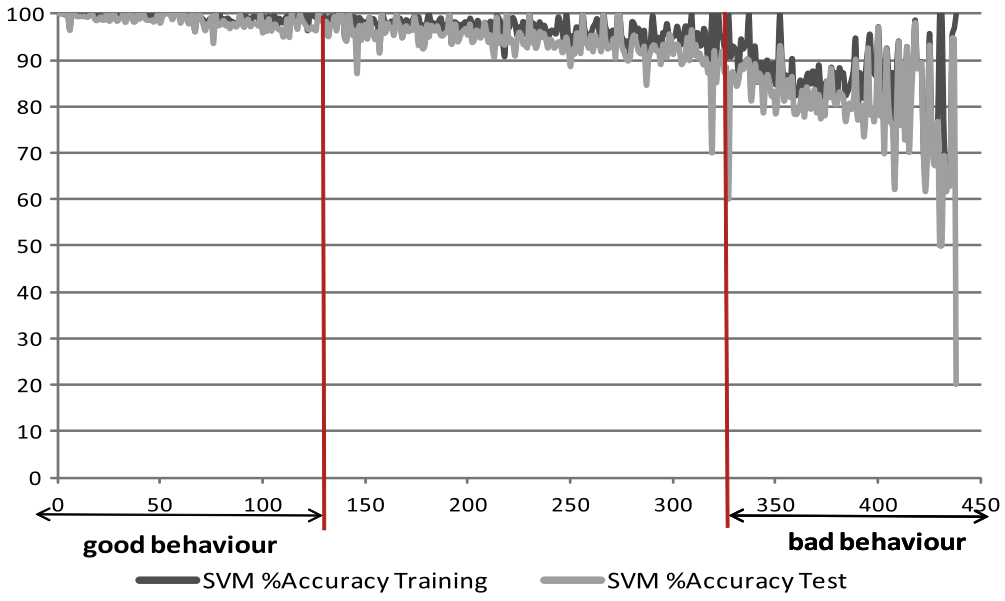

Fig. 10. SVM accuracy in training/test sorted by N1.

- A high value of N1 results in a bad performance of all the learning models considered. Those problems with a high percentage of instances being close to other class instances are difficult to model by the considered learning methods.
- A high value of N3 results in a bad performance of all the learning models considered. When the boundaries of the two classes are getting closer, the performance of the learning model decreases significantly.

With this information is possible to know in advance when the problem will be difficult for the ANN or the SVM model calculating the N1 and N3 measures before building the model. Therefore, it could be possible to relate the *good* or *bad performance* consequents of the rules with the necessity of a more sophisticated topology of the net, or with the use of more complex kernels for the SVM. The contrary case would be also verified, saving time in the model configuration and construction. Please note that the accuracy values shown in Table 5 cannot be used to predict the accuracy of the method for other data sets.

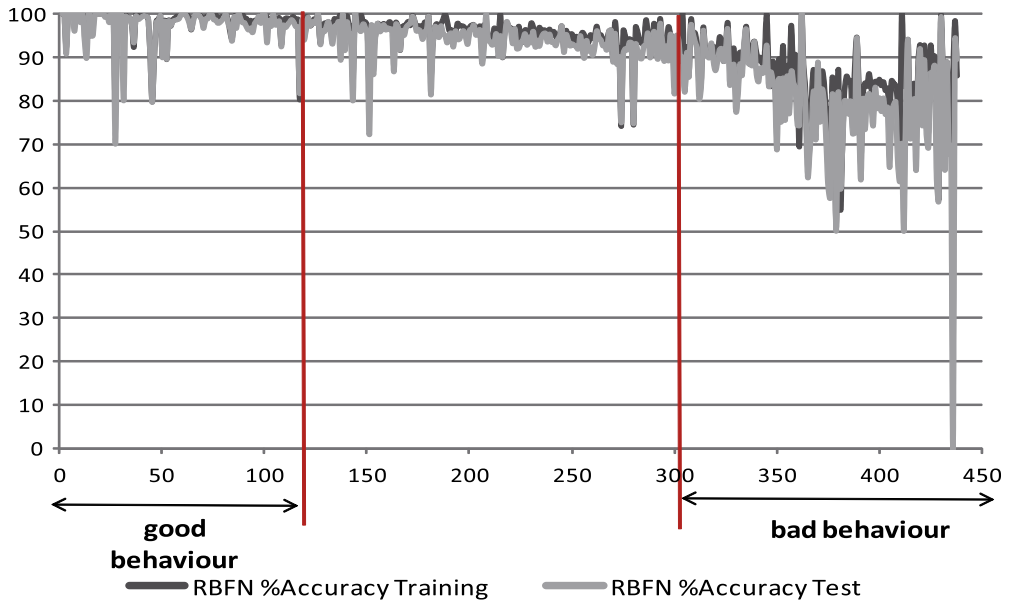

Fig. 11. RBFN accuracy in training/test sorted by N3.

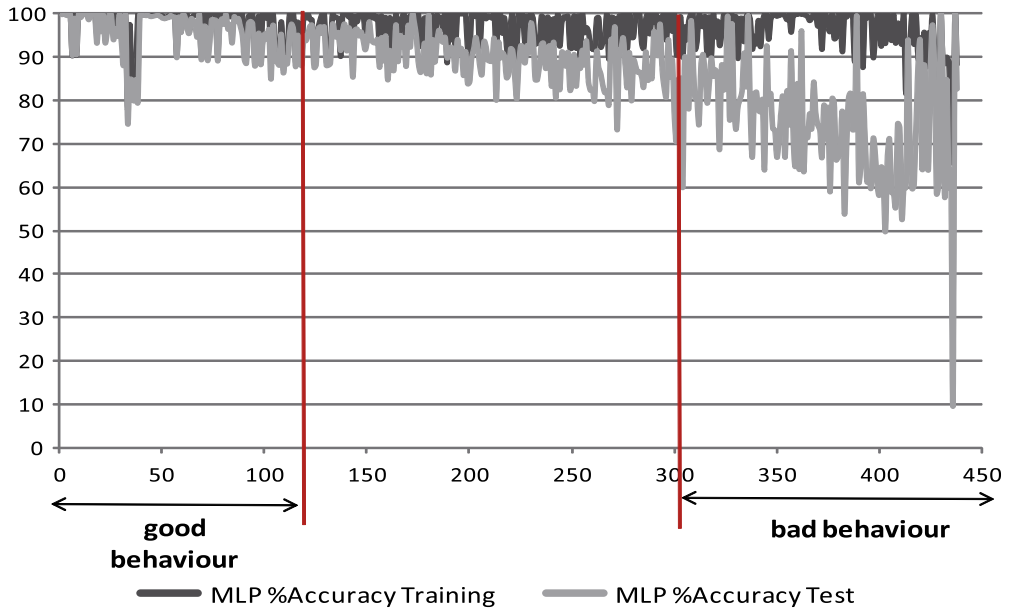

Fig. 12. MLP accuracy in training/test sorted by N3.

Although we have obtained some interesting individual rules, we can extend our study by considering the combination of these complexity metrics in order to obtain more interpretable, extensive and descriptive rules.

6.3. Collective evaluation of the set of rules

The objective of this section is to analyze the good and bad rules jointly. We have considered the disjunctive combination (we use the *or* operator) of all the positive rules to obtain a single rule (Positive Rule Disjunction-PRD-). The same procedure is performed with all the negative ones so we obtain another rule (Negative Rule Disjunction-NRD-). The new disjunctive rules will be activated if any of the component rules' antecedents are verified.

By means of merging the individual rules we can arrive at a more general description, with a wider support, of the performance of the learning models. In Table 6 we summarize both disjunctions, and a third rule representing those data sets which are not characterized by either one.

From the collective rules we can observe that the support has been increased from the single rules for PRD. NRD obtains similar support with respect to the single rules which compose it. Thus the geometrical aspects of the data that the N1 and N3 metrics are measuring are related, and the data sets included in the R1 and R2 individual rules show either beneficial

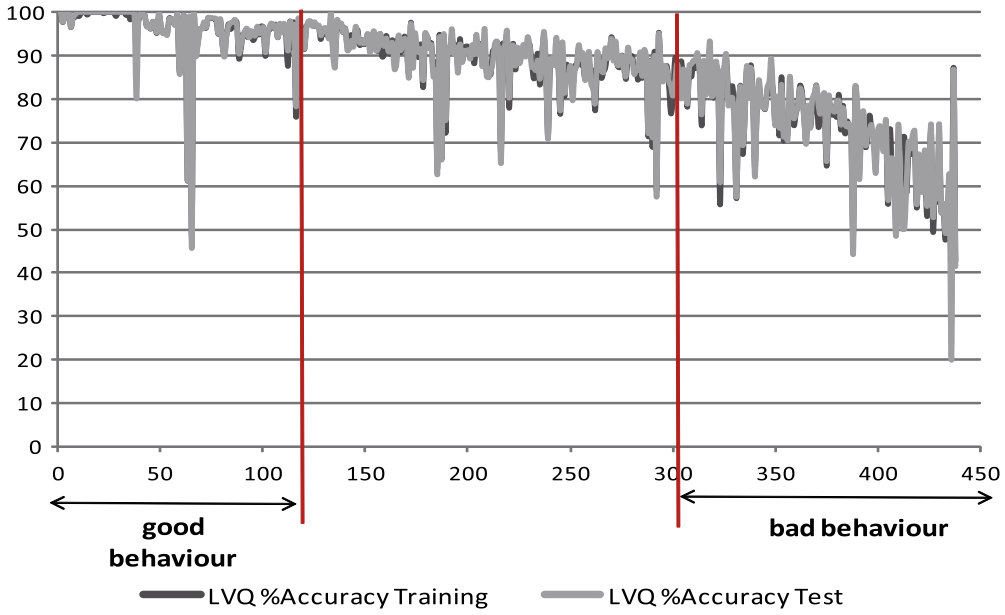

Fig. 13. LVQ accuracy in training/test sorted by N3.

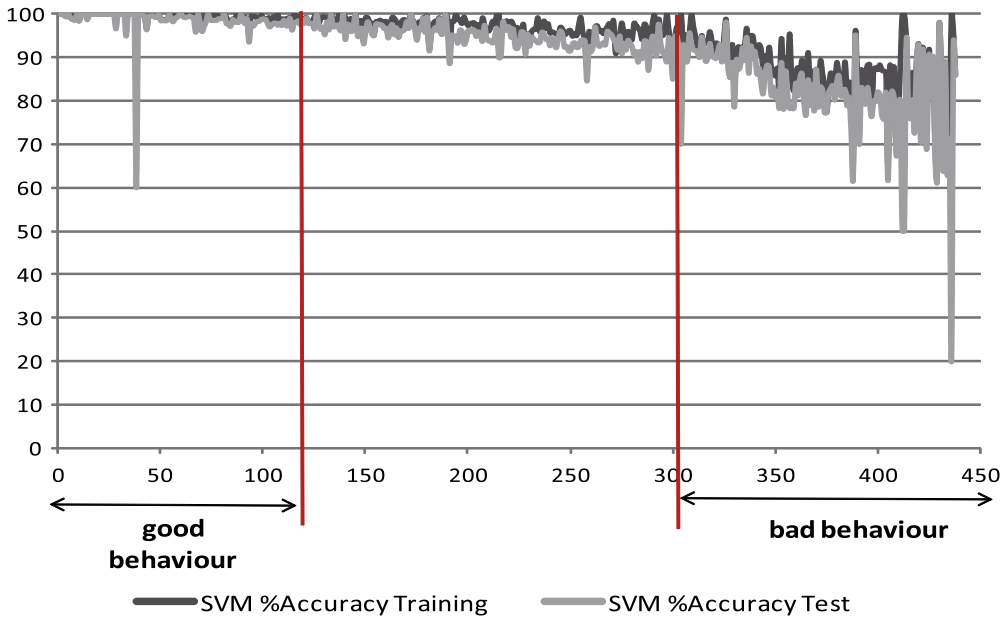

Fig. 14. SVM accuracy in training/test sorted by N3.

Table 4
Significant intervals.

Interval	ANNs performance
$N1 < 0.08$	Good performance
$N3 < 0.029$	Good performance
$N1 > 0.25$	Bad performance
$N3 > 0.108$	Bad performance

Table 5
Rules with one metric obtained from the intervals.

Id.	Rule	Support	Model	%Training	Training Diff. (%)	% Test	Test Diff. (%)
R1+	If N1[X] < 0.08 then good performance	29.22	RBFN	98.03	4.91	97.20	6.55
			MLP	97.72	1.69	95.95	7.56
			LVQ	95.65	10.26	95.86	10.47
			SVM	98.45	6.91	97.93	8.15
R2+	If N3[X] < 0.029 then good performance	26.71	RBFN	98.18	5.06	97.28	6.63
			MLP	97.31	1.33	96.02	7.63
			LVQ	95.67	10.28	95.83	10.44
			SVM	98.66	7.12	98.09	8.31
R1-	If N1[X] > 0.25 then good performance	24.43	RBFN	83.64	−9.48	78.22	−12.43
			MLP	93.68	−2.30	76.74	−11.65
			LVQ	70.53	−14.86	70.02	−15.37
			SVM	80.38	−11.16	76.81	−12.97
R2-	If N3[X] > 0.108 then bad performance	31.51	RBFN	85.33	−7.79	80.64	−10.01
			MLP	93.80	−2.18	78.47	−9.92
			LVQ	72.87	−12.52	72.63	−12.76
			SVM	82.34	−9.20	79.03	−10.75

Table 6
Disjunctive rules obtained from single rules.

Id.	Rule	Support	Model	% Training	Training diff. (%)	% Test	Test diff. (%)
PRD	If R1+ or R2+ then good performance		RBFN	98.16	5.04	97.11	6.46
		32.65	MLP	97.29	0.47	95.17	8.15
			LVQ	95.52	9.08	95.58	9.23
			SMO	99.27	7.00	98.30	8.12
NRD	If R1-or R2-then bad performance		RBFN	85.50	−7.62	80.81	−9.84
		31.96	MLP	96.12	−0.70	75.68	−11.34
			LVQ	74.04	−12.40	73.73	−12.62
			SMO	87.67	−8.99	82.68	−10.59
Not characterized	If not PRD and not NRD then good performance		RBFN	95.35	2.23	93.59	2.94
		35.39	MLP	97.03	0.21	89.74	2.72
			LVQ	89.26	2.82	89.24	2.89
			SMO	96.89	1.64	94.57	2.08

(R1+ and R2+) or harmful (R1-and R2-) characteristics for the ANNs and the SVM together. However, the values of both N1 and N3 measures cannot be directly correlated, as the data set ordering within each interval is different.

It is also important to notice that the test and training accuracy differences are maintained with respect to the single rules from Table 5 despite the support increment. In the case of MLP, the test differences for PRD are slightly greater than the test differences found for the individual R1+ and R2+ rules, while the training accuracy differences are close to zero, as found for the individual rules. In the case of LVQ the PRD rule decreases the test difference in 1%, while RBFN and SVM maintain the same values. In overall, the training and test differences for the PRD rule are the same than the individual rules with a 1% margin. MLP is the only method that show a similar training accuracy no matter the data set considered, but its performance depending on the data set can be seen in the test accuracy.

The absolute value of the test differences for the NRD rule are lower that the R1-individual rule due to the 7% of increased support. Both R2-and NRD rule have similar support, and the test differences are close as well. If we analyze the training differences, the R1-rule shows greater differences than the R2-one. Therefore we can conclude that the R1-rule characterize very harmful data sets for the ANNs and the SVM, while the R2-rule adds other ones which are less detrimental. Since no data sets are shared by PRD and NRD, we can consider three blocks of data sets with their respective support, as depicted in Figs. 15–18 (with no particular data set order within each block):

- The first block (the left-side one) represents the data sets covered by the PRD rule. They are the data sets recognized as being those in which the ANNs and the SVM have good performance.
- The second block (the middle one) plots the data sets for the NRD rule, which are bad data sets for the methods considered.
- The third and last block (the right-side one) contains the unclassified data sets by the previous two rules.

We can see that almost the 65% of the analyzed data sets are covered by these two rules with significant differences. Hence the PRD and NRD rules can be used to define the shared domains of competence of the ANNs and SVM models.

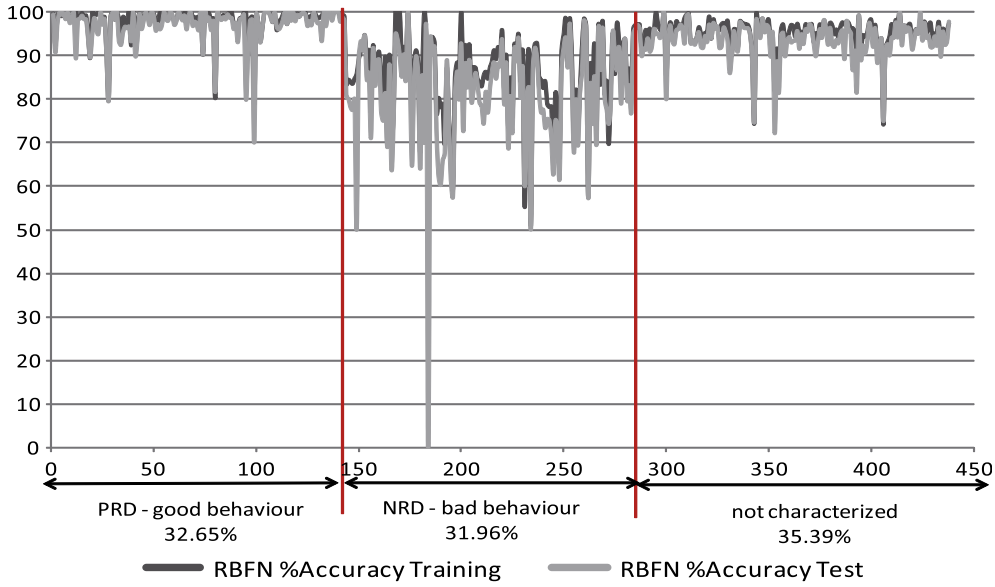

Fig. 15. Three blocks representation for PRD, NRD and not covered data sets for RBFN.

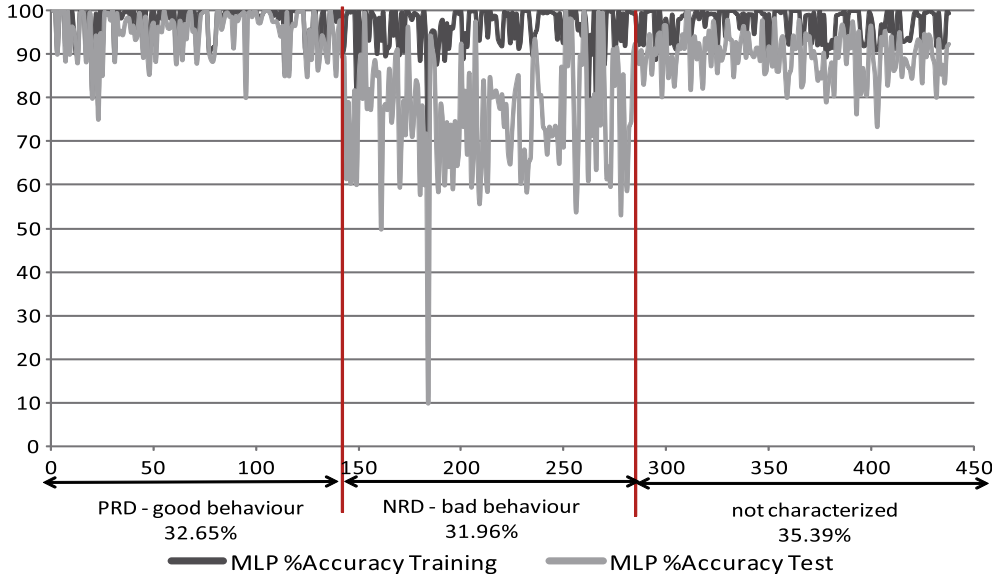

Fig. 16. Three blocks representation for PRD, NRD and not covered data sets for MLP.

6.4. Validation of the domains of competence

In this section the domains of competence obtained by means of the PRD and NRD rules are validated by means of a fresh bunch of 200 data sets. In Table 7 we have summarized the average training and test accuracy values for these new data sets (using the same parameters as before).

Table 8 presents the PRD and NRD rules average training and test accuracy for the validation data sets for each learning model, as well as the difference from the global average depicted in Table 7. The results depicted show the obtained differentiation between the good and bad performance with the rules PRD and NRD respectively. Please note that the values in the columns are now calculated using the new 200 data sets that fall in the interval of the correspondent rule, independently of the 438 initial ones. Thus the values in the columns "% Training" and "% Test" differ from those shown in Table 6, while the training and specially the test differences are similar, with less than a 2% difference in the worst case, supporting the characterization made for the ANNs and the SVM.

The three blocks figure representation of the data of Table 8 for the MLP, RBFN, LVQ and SVM methods are depicted in Figs. 19–22, respectively.

From these results we can observe the following:

- The bad performance of the classifiers which is characterized by NRD is maintained in this new set of data sets. The data sets covered by this rule show a bad performance for the learning methods considered.
- The PRD rule covers the new data sets for which the four learning methods perform remarkably well. Only in the case of LVQ appear some isolated outliers with test values below the 70%.
- The differences for these two rules for the four models in training and test are similar to those obtained in the previous section when extracting the characterization.
- The support of the PRD and NRD rules are similar to the "training" case. The support of the not characterized region is also similar.
- The test accuracy differences of the not characterized data sets are positive, but clearly below than the obtained by the PRD rule.

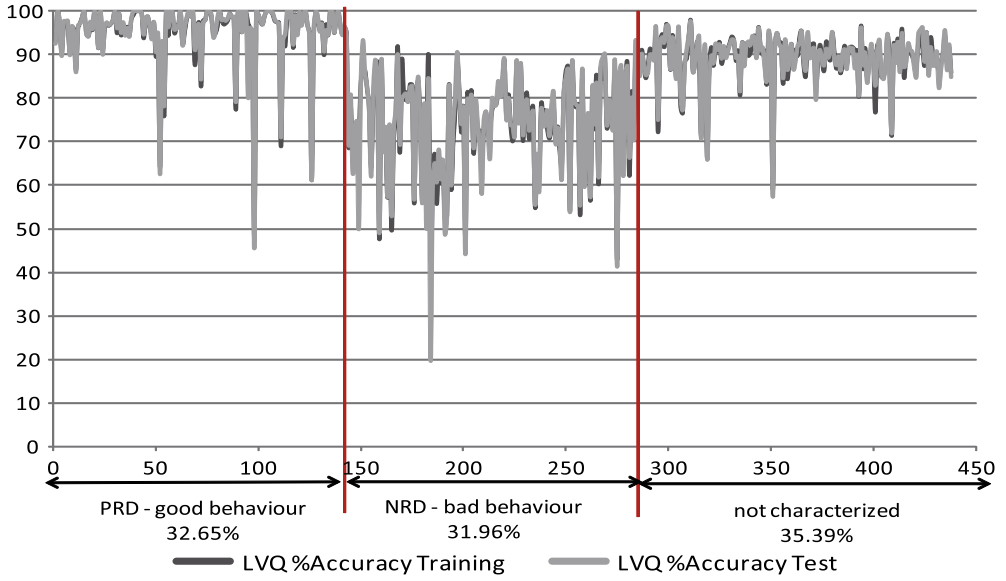

Fig. 17. Three blocks representation for PRD, NRD and not covered data sets for LVQ.

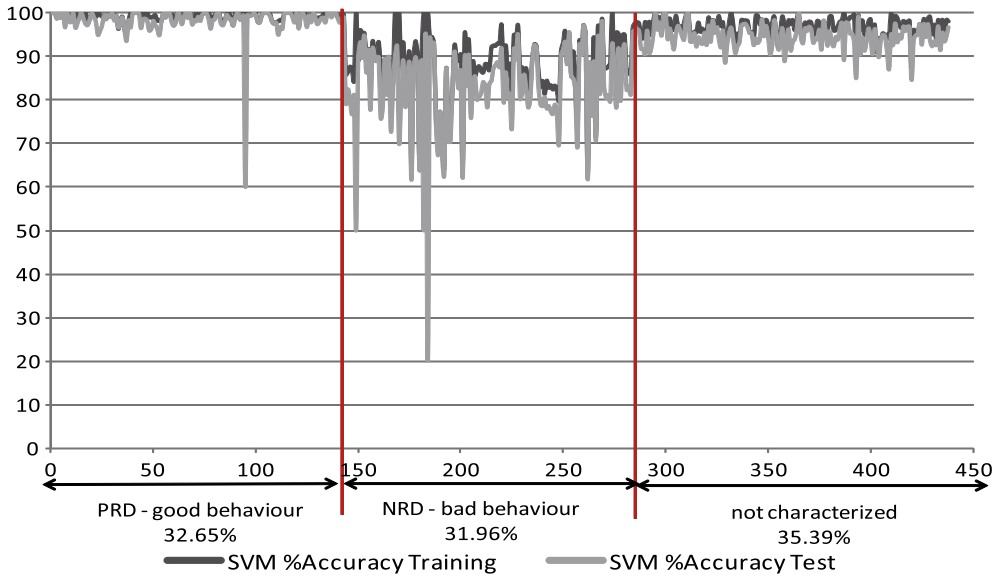

Fig. 18. Three blocks representation for PRD, NRD and not covered data sets for SVM.

Table 7
Global average training and test Accuracy/std. dev. for RBFN, MLP, LVQ and SVM over the validation data sets.

	Global % accuracy training global training std. dev.	Global % accuracy test global test std. dev.
RBFN	85.23% 11.51	84.84% 11.57
MLP	93.09% 6.90	88.89% 9.11
LVQ	83.12% 13.56	83.28% 13.63
SVM	91.57% 9.11	90.35% 9.77

Table 8
Validation results for PRD and NRD rules.

Id.	Rule	Support	Model	% Training	Training diff. (%)	% Test	Test diff. (%)
PRD	If R1+ or R2+ then good performance	34.50	RBFN	92.88	7.65	92.82	7.98
			MLP	96.50	3.41	95.33	6.44
			LVQ	93.63	10.51	93.71	10.43
			SMO	97.96	7.00	97.70	8.12
NRD	If R1-or R2-then bad performance	37.50	RBFN	76.29	−8.94	75.48	−9.36
			MLP	88.01	−5.08	80.28	−8.61
			LVQ	69.79	−13.33	69.98	−13.30
			SMO	83.38	−8.99	80.82	−10.59
Not characterized	If not PRD and not NRD then good performance	28.00	RBFN	87.78	2.55	87.53	2.69
			MLP	95.71	2.62	92.46	3.57
			LVQ	88.02	4.90	88.23	4.95
			SMO	94.67	1.64	94.04	2.08

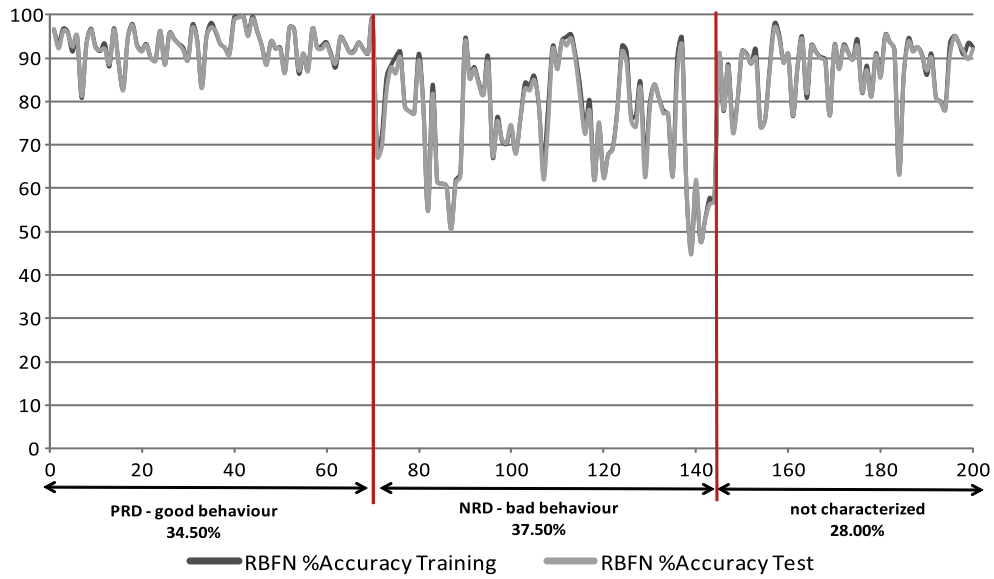

Fig. 19. Three blocks representation for PRD, NRD and not covered data sets for RBFN considering validation data sets.

From these statements, it is possible to conclude that the obtained domains of competence and their interpretation can indicate the characteristics of the data sets for which the ANNs and the SVM models considered would perform well or badly.

7. Lessons learned

This section is dedicated to provide some considerations on the use and information gained by the shared domains of competence presented in this paper. Their characteristics as well as suggestions on some of their aspects and details are enumerated:

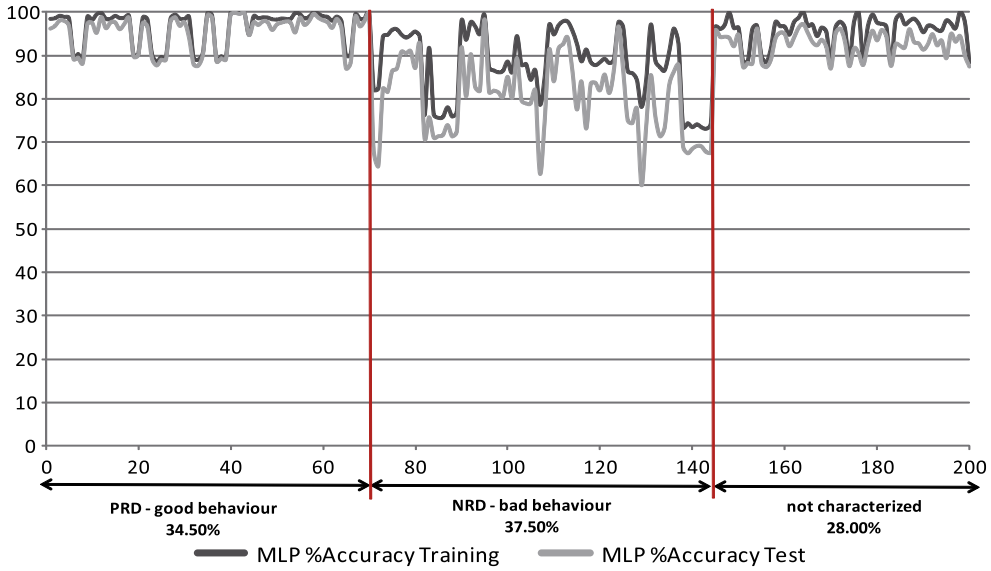

Fig. 20. Three blocks representation for PRD, NRD and not covered data sets for MLP considering validation data sets.

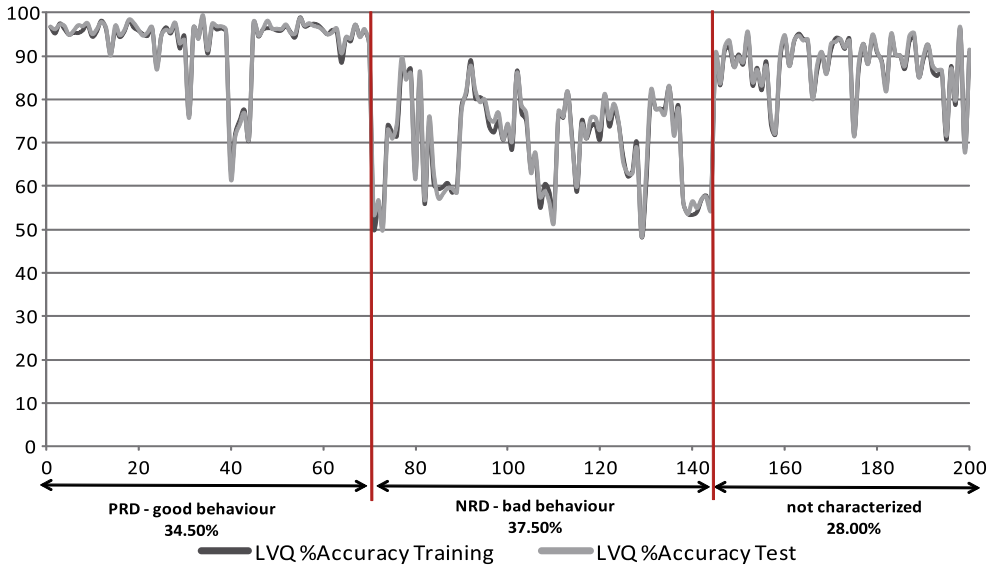

Fig. 21. Three blocks representation for PRD, NRD and not covered data sets for LVQ considering validation data sets.

- The use of the domains of competence cannot be used to predict concrete accuracy values of the classifiers. As can be seen from the validation results, the test accuracy obtained differs in some case in more than 5%. However, the performance of the methods are noticeably worse for those data sets included in the NRD rule with respect to those included in the PRD rule. They cannot be used to compare the performance between two methods, as we have indicated in Section 4.2.
- While the support obtained for the PRD rule indicates that the data sets covered by the good R1+ and R2+ individual rules are shared to a certain degree, the ordering within each interval differs. Some data sets can be found in R1+ but not in R2+ and vice versa. This is also true for the bad R1-and R2-rules with respect to the NRD rule. Although both N1 and N3 measures can be used to identify easy or hard problems for the four approximate learning methods considered, each one provides different information.
- Low values of the N1 measure are beneficial for the performance of the ANNs and the SVM. That means that these classifiers take advantage of data distributed in such a way that the percentage of the examples are not in the boundaries. That is, concentrated clusters of examples are better suited for the ANNs and SVM instead of large boundaries with a high proportion of data.
- Low values of the N3 measure are beneficial for all the four classifiers as well. That means that the presence of large gaps of the class boundaries facilitate the model generalization. This is evident in the case of the SVM algorithm, which tries to create a separating hyperplane, and lower values of the N3 measure are beneficial to this task.

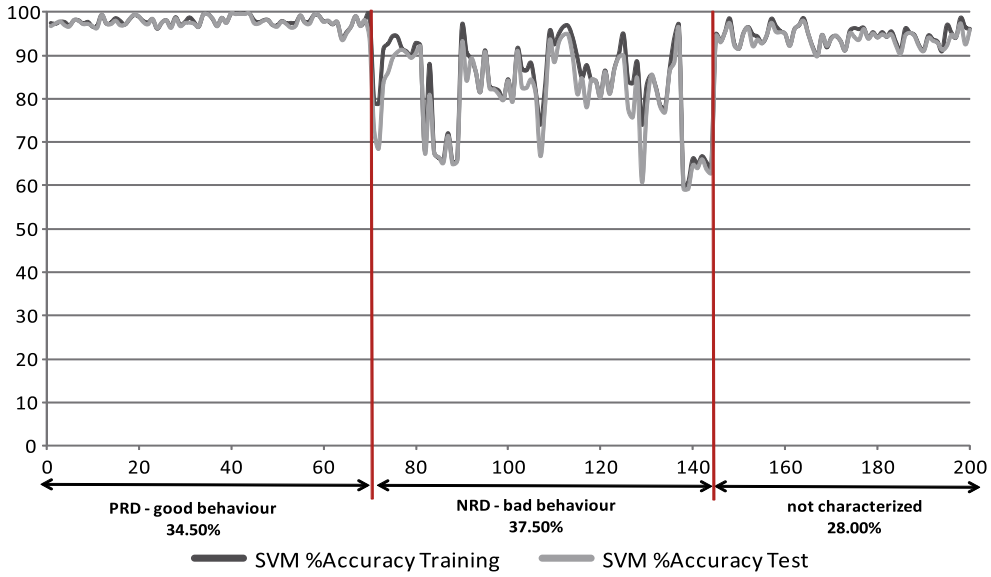

Fig. 22. Three blocks representation for PRD, NRD and not covered data sets for SVM considering validation data sets.

- High values of the N1 and N3 measures harm the performance of the ANNs and the SVM. In this case an alternative classifier can be considered if its domains of competence are known and suitable, or the use of a more complex topology (in the case of the ANNs) or SVM configuration (increasing the degree in a polynomial kernel for example) in order to obtain more sophisticate decision boundaries. A high N1 or N3 value could mean that these configurations must be taken into account instead of discouraging the use of the ANNs and SVM algorithms.
- In order to use an alternative classifier, its domains of competence must be known. They can be extracted following the approach described in [23] as carried out in this paper. This can be exhaustive, but, as we have demonstrated, related classifiers share wide regions of the domains of competence. Thus obtaining them for a limited number of representative classifiers would be enough to have a pool of classifiers to choose from.
- Finally we want to remark that not all the measures of separability of classes are informative for the four classifiers considered. As seen in Table 3, there are other measures that can provide information of the performance for a single ANN or the SVM. That means that there are aspects of the data in addition to the ones characterized by N1 and N3 that have influence in the classifier's performance. This should be taken into account if one of the classifiers studied is used individually.

8. Concluding remarks

We have performed a study over a set of binary data sets with three ANN methods and one SVM model. We have computed five data complexity measures for the data sets known as *measures of separability of classes* in order to obtain intervals of such metrics in which the method's performance is significantly good or bad. We have obtained descriptive rules for two measures, and studied the interaction between them.

We have obtained two final rules which are simple, interpretable and precise to describe the common good and bad performance of the ANNs and the SVM models considered in this work, thus establishing their shared domains of competence. These domains of competence have been validated using an extra amount of data sets, observing that they generalize well and describe the performance of the four models appropriately. Furthermore, we present the possibility of determining for which data sets RBFN, MLP, LVQ and SVM will perform well or badly prior to their execution using the obtained domains of competence.

We must point out that this is a particular study for four specific methods. On the other hand, this work presents a new challenge that could be extended to other ANN models or SVM methods, to analyze the relation between their domains of competence and parameter adjustment, and to develop new measures which could give more information on the performances of ANNs and SVMs for classification.

References

[1] M. Asaduzzaman, M. Shahjahan, K. Murase, Faster training using fusion of activation functions for feed forward neural networks, International Journal of Neural Systems 19 (2009) 437–448.
[2] A. Asuncion, D. Newman, UCI machine learning repository, 2007.
[3] M. Basu, T.K. Ho, Data Complexity in Pattern Recognition, Advanced Information and Knowledge Processing, Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2006.

[4] R. Baumgartner, R.L. Somorjai, Data complexity assessment in undersampled classification of high-dimensional biomedical data, Pattern Recognition Letters 12 (2006) 1383–1389.
[5] E. Bernadó-Mansilla, T.K. Ho, Domain of competence of XCS classifier system in complexity measurement space, IEEE Transactions on Evolutionary Computation 9 (2005) 82–104.
[6] J.C. Bezdek, L. Kuncheva, Nearest prototype classifier designs: an experimental study, International Journal of Intelligent Systems 16 (2001) 1445–1473.
[7] P. Brazdil, C. Giraud-Carrier, C. Soares, R. Vilalta, Metalearning: Applications to Data Mining, Cognitive Technologies, Springer, 2009.
[8] D. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.
[9] M. Buhmann, Radial Basis Functions: Theory and Implementations, Cambridge Monographs on Applied and Computational Mathematics (2003).
[10] R. Capparuccia, R. De Leone, E. Marchitto, Integrating support vector machines and neural networks, Neural Networks 20 (2007) 590–597.
[11] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.
[12] G. Daqi, L. Chunxia, Y. Yunfan, Task decomposition and modular single-hidden-layer perceptron classifiers for multi-class learning problems, Pattern Recognition 40 (2007) 2226–2236.
[13] M. Dong, R. Kothari, Feature subset selection using a new definition of classificabilty, Pattern Recognition Letters 24 (2003) 1215–1225.
[14] D. Du, K. Li, M. Fei, A fast multi-output rbf neural network construction method, Neurocomputing 73 (2010) 2196–2202.
[15] D. Fisch, B. Kühbeck, B. Sick, S.J. Ovaska, So near and yet so far: new insight into properties of some well-known classifier paradigms, Information Sciences 180 (2010) 3381–3401.
[16] S. García, J.R. Cano, E. Bernadó-Mansilla, F. Herrera, Diagnose of effective evolutionary prototype selection using an overlapping measure, International Journal of Pattern Recognition and Artificial Intelligence 23 (2009) 2378–2398.
[17] A. Ghosh, M. Biehl, B. Hammer, Performance analysis of lvq algorithms: a statistical physics approach, Neural Networks 19 (2006) 817–829.
[18] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 289–300.
[19] A. Kalousis, Algorithm selection via meta-learning, Ph.D. thesis, Université de Geneve, 2002.
[20] S.W. Kim, B.J. Oommen, On using prototype reduction schemes to enhance the computation of volume-based inter-class overlap measures, Pattern Recognition 42 (2009) 2695–2704.
[21] Y. Li, M. Dong, R. Kothari, Classifiability-based omnivariate decision trees, IEEE Transactions on Neural Networks 16 (2005) 1547–1560.
[22] J. Luengo, S. García, F. Herrera, A study on the use of statistical tests for experimentation with neural networks: analysis of parametric test conditions and non-parametric tests, Expert Systems with Applications 36 (2009) 7798–7808.
[23] J. Luengo, F. Herrera, Domains of competence of fuzzy rule based classification systems with data complexity measures: a case of study using a fuzzy hybrid genetic based machine learning method, Fuzzy Sets and Systems 161 (2010) 3–19.
[24] S. Maldonado, R. Weber, J. Basak, Simultaneous feature selection and classification using kernel-penalized support vector machines, Information Sciences 181 (2010) 115–128.
[25] M.F. Moller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.
[26] R.A. Mollineda, J.S. Sánchez, J.M. Sotoca, Data characterization for effective prototype selection, in: Proceedings of the 2nd Iberian Conference on Pattern Recognition and Image Analysis, Springer, 2005, pp. 27–34.
[27] Y.J. Oyang, S.C. Hwang, Y.Y. Ou, C.Y. Chen, Z.W. Chen, Data classification with radial basis function networks based on a novel kernel density estimation algorithm, IEEE Transactions on Neural Networks 16 (2005) 225–236.
[28] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.
[29] M.J. Powell, Algorithm for approximation, Clarendon Press, Oxford, 1987. pp. 143–168.
[30] C. Renjifo, D. Barsic, C. Carmen, K. Norman, G.S. Peacock, Improving radial basis function kernel classification through incremental learning and automatic parameter selection, Neurocomputing 72 (2008) 3–14.
[31] R. Rojas, Neural Networks: A Systematic Introduction, Springer, 1996.
[32] J.S. Sánchez, R.A. Mollineda, J.M. Sotoca, An analysis of how training data complexity affects the nearest neighbor classifiers, Pattern Analysis & Applications 10 (2007) 189–201.
[33] S. Singh, Multiresolution estimates of classification complexity, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 1534–1539.
[34] F.W. Smith, Pattern classifier design by linear programming, IEEE Transactions on Computers 17 (1968) 367–372.
[35] S. Suresh, N. Sundararajan, P. Saratchandran, Risk-sensitive loss functions for sparse multi-category classification problems, Information Sciences 178 (2008) 2621–2638.
[36] Q. Tao, D. Chu, J. Wang, Recursive support vector machines for dimensionality reduction, IEEE Transactions on Neural Networks 19 (2008) 189–193.
[37] T. Villmann, B. Hammer, F.M. Schleif, W. Hermann, M. Cottrell, Fuzzy classification using information theoretic learning vector quantization, Neurocomputing 71 (2008) 3070–3076.
[38] Z. Xu, K. Huang, J. Zhu, I. King, M. Lyu, A novel kernel-based maximum a posteriori classification method, Neural Networks 22 (2009) 977–987.
[39] C.Y. Yang, J.S. Yang, J.J. Wang, Margin calibration in svm class-imbalanced learning, Neurocomputing 73 (2009) 397–411.
[40] S.S. Yang, S. Siu, C.L. Ho, Analysis of the initial values in split-complex backpropagation algorithm, IEEE Transactions on Neural Networks 19 (2008) 1564–1573.