Detecting relevant variables and interactions for classification in Support Vector Machines

Emilio Carrizosa

Universidad de Sevilla (Spain). ecarrizosa@us.es

Belén Martín-Barragán

Universidad de Sevilla (Spain). belmart@us.es

Dolores Romero Morales

University of Oxford (United Kingdom). dolores.romero-morales@sbs.ox.ac.uk

April 12, 2006

Abstract

The widely used Support Vector Machine (SVM) method has shown to yield good results in Supervised Classification problems. The Binarized SVM (BSVM) is a variant which is able to automatically detect which variables are, by themselves, most relevant for the classifier. In this work, we extend the BSVM introduced by the authors to a method that, apart from detecting the relevant variables, also detects the most relevant interactions between them. The classification ability of the proposed method is comparable to standard SVM for different kernels and clearly better than Classification Trees. Our method involves the optimization of a Linear Programming problem with a large number of decision variables, for which we use the well-known Column Generation technique.

Keywords: Data Mining, Supervised Classification, Column Generation, Support Vector Machines, Interactions Detection.

1 Introduction and literature review

Classifying objects or individuals into different classes or groups is one of the aims of Data Mining, [6]. This topic has been addressed in different areas such as Statistics, Operations Research and Artificial Intelligence. We focus on the well-known so-called Supervised Classification problem, usually referred as Discriminant Analysis by statisticians.

We have a set of objects Ω, partitioned into a set of classes C. The aim is to build a classification rule which predicts the class membership $c^u \in \mathcal{C}$ of an object $u \in \Omega$, by means of its *predictor vector* x^u. The predictor vector x^u takes values in a set X which is usually assumed to be a subset of $\mathbb{R}^p$, and the components x_ℓ, $\ell = 1, 2, \ldots, p$, of the predictor vector x are called *predictor variables*.

Not all the information about the objects in Ω is available, but only in a subset I, called the *training sample*, where both predictor vector and class-membership of the objects are known. With this information, the classification rule must be built.

A popular and powerful tool for Supervised Classification are the so-called Support Vector Machines (SVM) [3], which consist in finding the hyperplane with maximal margin, i.e., the one furthest from the closest object. The most popular versions of SVM embed, via a so-called kernel function, the original predictor variables into a higher (possibly infinite) dimensional space, [8]. In this way one obtains classifiers with good generalization properties. However, it is hard to find out, in the obtained rules, which variables are more relevant for the classification and how interactions between them affect the classifier.

In [2], the authors have introduced the so-called Binarized Support Vector Machine (BSVM) method: each variable is replaced by a set of binary variables obtained by queries of the following type:

$$\boxed{\text{is predictor variable } \ell \text{ greater than or equal to } b?} \tag{1}$$

The classification rules obtained this way have a classification behavior comparable to the standard linear SVM and clearly better than Classification Trees, [2]. Although BSVM gives a powerful tool for detecting which are the most relevant variables, interactions between them are not taken into account. In

this paper we extend the BSVM in order to get classifiers that detect the interactions between variables which are useful to improve the classification performance. We call the proposed extension Non-linear Binarized Support Vector Machine (NBSVM).

In addition to considering queries of type (1), for all possible cutoffs b, NBSVM also considers the simultaneous positive answer to two or more of such queries, i.e.,

Is predictor variable ℓ_1 greater than or equal to b_1,
and predictor variable ℓ_2 greater than or equal to b_2,
$\vdots$
and predictor variable ℓ_g greater than or equal to b_g?

(2)

Special attention will be paid to the case $g = 2$, in which the pairwise interaction is explored by queries of type

Is predictor variable ℓ_1 greater than or equal to b_1,
and predictor variable ℓ_2 greater than or equal to b_2?

(3)

For this particular case, it is easy to measure how the classifier is affected by the interaction of each pair of variables, as shown in Section 2.

We restrict ourselves to the case in which two classes exist, $\mathcal{C} = \{-1, 1\}$. The multiclass case can be reduced to a series of two-class problems, as has been suggested e.g. in [7, 8, 16].

Numerical results show that the NBSVM gives a classifier that behaves comparable to standard SVM for different choices of the kernels, and clearly better than Classification Trees. Moreover, the tradeoff between interpretability and classification ability can be controlled via an upper bound on the number of features allowed. At the same time, the method provides a very powerful tool for detecting which interactions are relevant for classication.

The remainder of the paper is organized as follows: NBSVM is described in Section 2. Since the number of features to be considered may be huge, the NBSVM method yields an optimization problem with a large number of decision variables, in general, of order $(\sharp(I)p)^g$, where $\sharp(\cdot)$ denotes the cardinality

of a set. The classifier uses the weighted sum of features defined from queries of type (2) and an independent term. In Section 3, a mathematical program for the choice of the weights is formulated and the Column-Generation-based algorithm proposed in [2] is extended in order to solve such a program when interactions between predictor variables are considered. Numerical results are shown in Section 4, whereas conclusions and some lines for future research are discussed in Section 5.

2 Binarizing the variables and their interactions

In practical applications, classification accuracy of the obtained classifier is not the only concern, but other characteristics of the classifier are also taken into account. For instance, in microarray analysis, interpretability is one of the issues that influences the choice of a prediction method, [13], where easily interpretable models, which might help to provide new medical insights, are sometimes preferred. In some fields, as diverse as cancer diagnosis and credit scoring, doctors or lenders might find important to easily explain the classification rule and detecting which combinations of variables are critical to predict class membership.

In practical Data Mining applications, quantitative variables usually appear together with the qualitative ones. In order to deal with continuous variables in the form of the presence or absence of certain symptom, rules of type (1) are used, where the cutoff value b needs to be chosen. For example, a doctor would say that having high blood pressure is a symptom of disease. Choosing the threshold b from which a specific blood pressure would be considered high is not usually an easy task.

In [2], all the possible rules of type (1) are theoretically considered, mathematically formalized by the function

$$\phi_{\ell b}(x) = \begin{cases} 1 & \text{if } x_\ell \geq b \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

for $b \in B_\ell = \{x_\ell^u : u \in I\}$ and $\ell = 1, 2, \ldots, p$.

The set of possible cutoff values b in (4) could be, in principle, $\mathbb{R}$. Thus the number of function of type (4) would, in principle, be infinite. However, given a training sample I, many of those possible cutoffs

will yield exactly the same classification in the objects in I, which are the objects whose information is available. In this sense, for a certain predictor variable ℓ and a given training sample I, we can constrain the choice of $b \in I\!R$ to the finite set B_ℓ.

Note that binary 0-1 variables can be accommodated to this framework easily, by taking $b = 1$. It might be less obvious for ordinal variables, i.e. qualitative variables whose values can be sorted according to some meaningful order $\succeq$. For instance, a variable ℓ taking the values {'big', 'medium', 'small'} yields the following queries of type (1): "Is $x_\ell \succeq$ big?", with affirmative answer for $x_\ell \in$ {'big'}; "Is $x_\ell \succeq$ medium?", with affirmative answer for $x_\ell \in$ {'medium', 'small'}; and "Is $x_\ell \succeq$ small?" always affirmatively answered.

For nominal variables, where there exists no meaningful order for the k values they take, queries of type (1) make no sense. In order to accommodate all types of variables to a common framework, a preprocessing step is needed where every nominal variable ℓ is replaced by k new variables as follows: for every possible value $\hat{x}$ of the original nominal variable ℓ, a new binary variable is built taking value one when x_ℓ is equal to $\hat{x}$ and zero otherwise. For instance, a variable ℓ taking values {'red','blue','green'} is replaced by three binary variables asking the questions 'is $x_\ell =$ red?', 'is $x_\ell =$ blue?' and 'is $x_\ell =$ green?'.

Example 1 *In the Credit Screening Database, from the UCI Machine Learning Repository [11], there are 15 variables, six of which are continuous (c), four binary (b) and five are nominal (n). Since no information about the meaning of the values is provided, we have considered that values cannot be sorted according to a meaningful order, and have encoded them as explained above. After the encoding, the original set of 15 variables, whose information about their names, types and the values they take, is given in Table 1, is replaced by a set of 43 variables.*

Example 2 *In the Cylinder Bands Database, from the UCI Machine Learning Repository [11], there are 35 variables (excluding the first four attributes, which are for identification of the object), twenty of which are considered to be continuous (c), five binaries (b), three are considered to be ordinal (o) and eight are are considered to be nominal (n). All objects having at least one missing value have been removed from*

name	type	values
A1	b	b, a
A2	c	[13.75,76.75]
A3	c	[0,28]
A4	n	u, y, l, t
A5	n	g, p, gg
A6	n	c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff
A7	n	v, h, bb, j, n, z, dd, ff, o
A8	c	[0,28.5]
A9	b	t, f
A10	b	t, f
A11	c	[0,67]
A12	b	t, f
A13	n	g, p, s
A14	c	[0,2000]
A15	c	[0,100000]

Table 1: Types of variables in Credit Screening Database.

the database. After the encoding, the original set of 34 variables, whose information about its name, type and values it takes is given in Table 2, is replaced by a set of 56 variables.

The family of functions trained with the encoding described above is given by

$$\hat{\mathcal{F}} = \{\phi_{\ell b} : b \in B_\ell, \ell = 1, 2, \ldots, p\},$$

and is used in BSVM, as described in [2]. In this paper we use a richer model, which takes into account the interactions between variables.

We define $\mathcal{F}$ as the set of all products of degree up to g of functions of $\hat{\mathcal{F}}$, i.e.,

$$\mathcal{F} = \left\{ \prod_{\phi \in F} \phi : F \subset \hat{\mathcal{F}}, \sharp(F) \leq g \right\}. \tag{5}$$

In what follows, each function of type $\phi \in \mathcal{F}$ is called *feature*. Moreover, features in $\hat{\mathcal{F}}$ are called *features of degree one* whereas a feature $\phi \in \mathcal{F}$, made up of the product of k features of degree one, $\phi = \phi_{\ell_1,b_1} \cdot \phi_{\ell_2,b_2} \ldots \phi_{\ell_k,b_k}$ with $\ell_i \neq \ell_j \forall i \neq j$ and $b_\ell \neq \min_{u \in I} x_\ell^u$, is said to be a feature of degree k, for all $k = 2, \ldots, g$.

The classification is performed as follows: each feature $\phi \in \mathcal{F}$ has associated a weight ω_ϕ, measuring its contribution for the classification into the class -1 or 1; the weighted sum of those features and a

name	type	values
grain screened	b	yes, no
ink color	b	key, type
proof on ctd ink	b	yes, no
blade mfg	n	benton, daetwyler, uddeholm
cylinder division	n	gallatin, warsaw, mattoon
paper type	n	uncoated, coated, super
ink type	n	uncoated, coated, cover
direct steam	b	yes, no
solvent type	n	xylol, lactol, naptha, line, other
type on cylinder	b	yes, no
press type	n	WoodHoe70, Motter70, Albert70, Motter94
press	o	802, 813, 815, 816, 821, 824, 827, 828
unit number	o	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
cylinder size	n	catalog, spiegel, tabloid
paper mill location	n	northUS, southUS, canadian, scandanavian, mideuropean
plating tank	b	1910, 1911
proof cut	c	[0,100]
viscosity	c	[0,100]
caliper	c	[0,1.0]
ink temperature	c	[5,30]
humifity	c	[5,120]
roughness	c	[0,2]
blade pressure	c	[10,75]
varnish pct	c	[0,100]
press speed	c	[0,4000]
ink pct	c	[0,100]
solvent pct	c	[0,100]
ESA Voltage	c	[0,16]
ESA Amperage	c	[0,10]
wax	c	[0,4.0]
hardener	c	[0,3.0]
roller durometer	c	[15,120]
current density	c	[20,50]
anode space ratio	c	[70,130]
chrome content	c	[80,120]

Table 2: Variables in Cylinder Bands Database.

threshold β constitute the *score function* f,

$$f(x) = \omega^\top \Phi(x) + \beta = \sum_{\phi \in \mathcal{F}} \omega_\phi \phi(x) + \beta, \tag{6}$$

where $\Phi(x) = (\phi(x))_{\phi \in \mathcal{F}}$ and $\omega^\top \Phi(x)$ denotes the scalar product of vectors ω and $\Phi(x)$.

Objects will be allocated to class -1 if $f(x) < 0$, and to class 1 if $f(x) > 0$. In case of ties, i.e. $f(x) = 0$, objects can be allocated randomly or by some predefined order. In this paper, following a worst case approach, they will be considered as misclassified.

Special attention will be paid to the case $g = 2$. For each pair of variables (ℓ_1, ℓ_2) let $\phi_{\ell_1,\ell_2,b_1,b_2} =$

$\phi_{\ell_1,b_1} \cdot \phi_{\ell_2,b_2}$. For simplicity in the notation, we denote $\omega_{\phi_{\ell_1,\ell_2,b_1,b_2}}$ and $\omega_{\phi_{\ell,b}}$ by $\omega_{\ell_1,\ell_2,b_1,b_2}$ and $\omega_{\ell,b}$, respectively. With this notation, the score function (6) can be rephrased as

$$f(x) = \sum_{\ell=1}^{p} \sum_{b \in B_\ell | x_\ell \geq b} \omega_{\ell b} + \sum_{\ell_1=1}^{p} \sum_{\ell_2=1}^{p} \sum_{b_2 \in B_{\ell_1} | x_{\ell_1} \geq b_1} \sum_{b_2 \in B_{\ell_2} | x_{\ell_2} \geq b_2} \omega_{\ell_1,\ell_2,b_1,b_2} + \beta. \tag{7}$$

The weight $\omega_{\ell b}$ associated to feature $\phi_{\ell b}$ represents the amount with which the query 'is $x_\ell \geq b$?' contributes to the score function (7). For a certain pair of variables (ℓ_1, ℓ_2), the coefficient $\omega_{\ell_1,\ell_2,b_1,b_2}$ represents the amount with which the query 'is $x_{\ell_1} \geq b_1$ and simultaneously $x_{\ell_2} \geq b_2$?' contributes to the score function (7). The role played in the score function (7) by the interaction between variables ℓ_1 and ℓ_2, is represented by the function $\varphi_{\ell_1,\ell_2}(s,t)$,

$$\varphi_{\ell_1,\ell_2}(s,t) = \sum_{b_2 \in B_{\ell_1} | s \geq b_1} \sum_{b_2 \in B_{\ell_2} | t \geq b_2} \omega_{\ell_1,\ell_2,b_1,b_2}, \quad \forall (s,t) \in I\!R^2.$$

Plotting ϕ_{ℓ_1,ℓ_2} gives a clear idea of how the interaction of ℓ_1 and ℓ_2 affects the classification.

Example 3 *Taking the Credit Screening Database, as in Example 1, and for ω obtained by the method explained in Section 3, function $\varphi_{\texttt{A14},\texttt{A3}}$, is plotted in Figure 1, where, at each point (s,t) in the graphic, the gray intensity represents the value of $\varphi_{\texttt{A14},\texttt{A3}}(s,t)$. The color in the down-left corner of the pictures corresponds to the null value, whereas lighter levels of gray corresponds to negative values and darker ones correspond to positive values.*

The same information is given in Table 3. For instance, an object having $x^u_{\texttt{A3}} = 3$ and $x^u_{\texttt{A14}} = 800$, would have $\varphi_{\texttt{A14},\texttt{A3}}(800,3) = \omega_{\texttt{A14},\texttt{A3},210,0.375} = 0.6422$, represented by the darkest gray area in Figure 1, whereas an object having $x^u_{\texttt{A3}} = 15$ and $x^u_{\texttt{A14}} = 500$, would have $\varphi_{\texttt{A14},\texttt{A3}}(400,15) = \omega_{\texttt{A14},\texttt{A3},210,0.375} + \omega_{\texttt{A14},\texttt{A3},232,5.835} + \omega_{\texttt{A14},\texttt{A3},70,9.5} = 0.6422 - 0.2613 - 1.0642 = -0.6833$, represented by the light gray area in the top-right corner of Figure 1.

The interaction of those pair of variables (ℓ_1, ℓ_2) having $\omega_{\ell_1,\ell_2,b_1,b_2} = 0$ for all $b_1 \in B_{\ell_1}$ and all $b_2 \in B_{\ell_2}$, has no effect in the classification based on score function (7). Moreover, the maximal absolute value that function φ takes in $I\!R \times I\!R$ measures how important for the classification is the interaction of the pair of

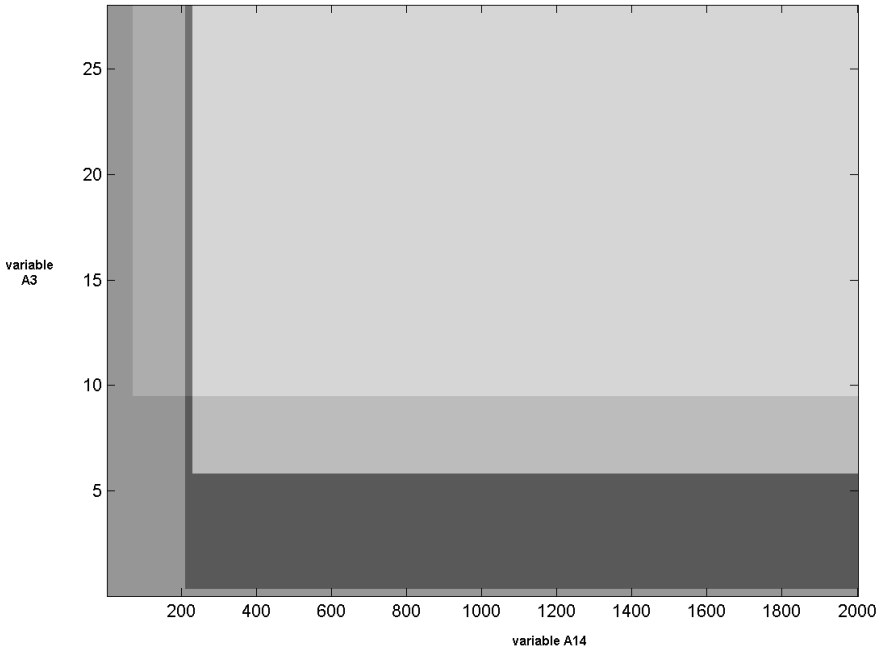

Figure 1: Role of the interaction in the score function. Database `credit`.

values of A3 \ values of A14	$[0, 70)$	$[70, 210)$	$[210, 232)$	$[232, 2000]$
$[0.000, 0.375)$	0	0	0	0
$[0.375, 5.835)$	0	0	0.6422	0.6422
$[5.835, 9.500)$	0	0	0.6422	-0.4220
$[9.500, 28.000)$	0	-0.2613	0.3809	-0.6833

Table 3: Role of the interaction in the score function.

variables (ℓ_1, ℓ_2). Such a measure, which we call *interaction intensity*, is defined, for a pair of variables (ℓ_1, ℓ_2) with $\ell_1 \neq \ell_2$, as follows

$$\begin{aligned} I(\ell_1, \ell_2) &= \max_{s,t \in I\!R} \left| \sum_{b_2 \in B_{\ell_1} | s \geq b_1} \sum_{b_2 \in B_{\ell_2} | t \geq b_2} \omega_{\ell_1, \ell_2, b_1, b_2} \right| \\ &= \max_{u \in \Omega} \left| \sum_{b_1 \in B_{\ell_1}} \sum_{b_2 \in B_{\ell_2}} \phi_{\ell_1, \ell_2, b_1, b_2}(x^u) \right|. \end{aligned}$$

Following a similar discussion, the importance for the classification of a variable ℓ, without taking into account its interactions with other variables, is given by

$$\begin{aligned} I(\ell, \ell) &= \max_{s \in I\!R} \left| \sum_{b \in B_\ell | s \geq b} \omega_{\ell, b} \right| \\ &= \max_{u \in \Omega} \left| \sum_{b \in B_\ell} \phi_{\ell, b}(x^u) \right|. \end{aligned}$$

Example 4 *For the Credit Screening Database, Figure 2 represents in a gray scale the different values of $I(\ell_1, \ell_2)$ for each pair of variables (ℓ_1, ℓ_2), for ω obtained by the method explained in Section 3. From this picture, it is evident that in the classifier, the most relevant variable is the continuous variable* A8 *and the pairs with highest interaction intensity I are:*

- *nominal variable* A5 *taking the value 'p', and continuous variable* A14
- *continuous variable* A2 *and nominal variable* A7 *taking the value 'v', and*
- *continuous variables* A2 *and* A3.

For many pairs of variables interactions are discarded by the classifier: for instance, the interaction between continuous variables A2 *and* A11*, or nominal variables* A5 *and* A6.

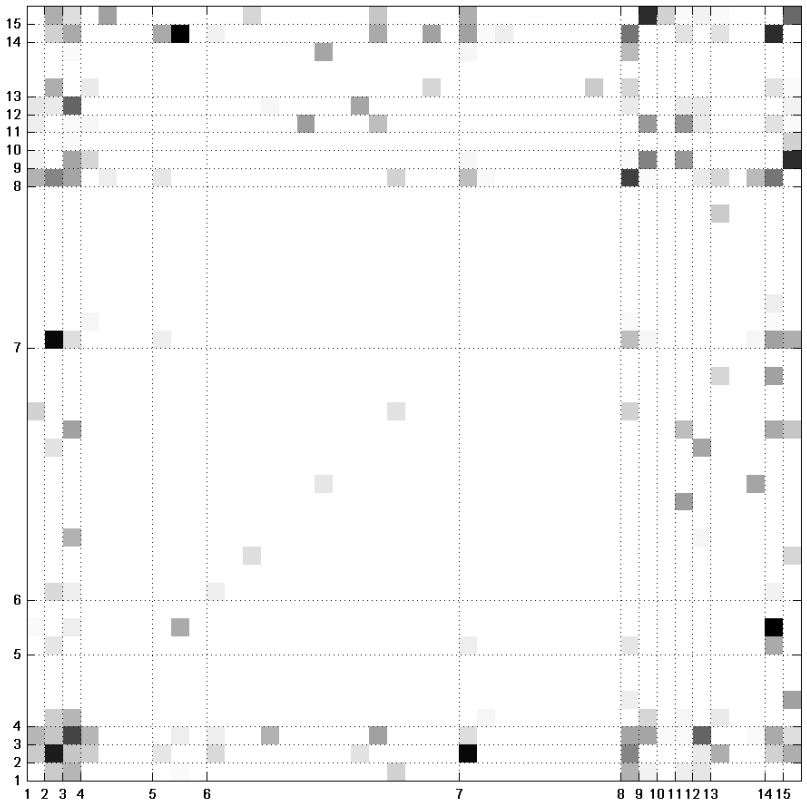

Figure 2: Interaction between pairs of variables. Database `credit`.

Example 5 *Taking the Cylinder Bands, whose variables are described in Example 2, Figure 4 represents, in a gray scale the different values of $I(\ell_1, \ell_2)$ for each pair of variables (ℓ_1, ℓ_2), obtained after applying the method that will be explained in Section 3. For the nine pairs of variables ℓ_1 and ℓ_2 for which $I(\ell_1, \ell_2)$ is highest, function φ, is plotted in a gray scale in Figure 3. Black (respectively white) colors correspond to the highest (respectively lowest) value of $I(\ell_1, \ell_2)$ for any pair of variables.*

For instance, looking at the graphic of variables `humifity` *and* `hardener` *and going from the bottom-left corner in the top-right direction, the gray intensity becomes lighter and lighter. This occurs because features associated to those variables have negative weights. In the graphic of variables* `ink pct` *and* `viscosity`*, two weights are positive and one negative.*

3 Building the classifier

In order to choose ω and β in (6) we follow, a soft-margin SVM-based approach [3], which consists in finding the hyperplane which maximizes the margin in the feature space, but allowing some objects to be misclassified.

The reminder of this Section is as follows: We first derive the Mathematical Programming formulation for the problem of maximizing the soft margin. Then, we describe in Sections 3.2 and 3.3 a Column Generation approach to solve the problem. We finally summarize the procedure and give some implementation details in Section 3.4.

3.1 Soft-margin mathematical formulations

The soft-margin maximization problem is formulated in this paper by

$$\begin{array}{lll} \min & \|\omega\| + C \sum_{u \in I} \xi^u & \\ s.t.: & c^u \left(\omega^\top \Phi(x^u) + \beta\right) + \xi^u \geq 1 & \forall u \in I \\ & \xi^u \geq 0 & \forall u \in I \\ & \omega \in I\!R^N, \beta \in I\!R, & \end{array} \tag{8}$$

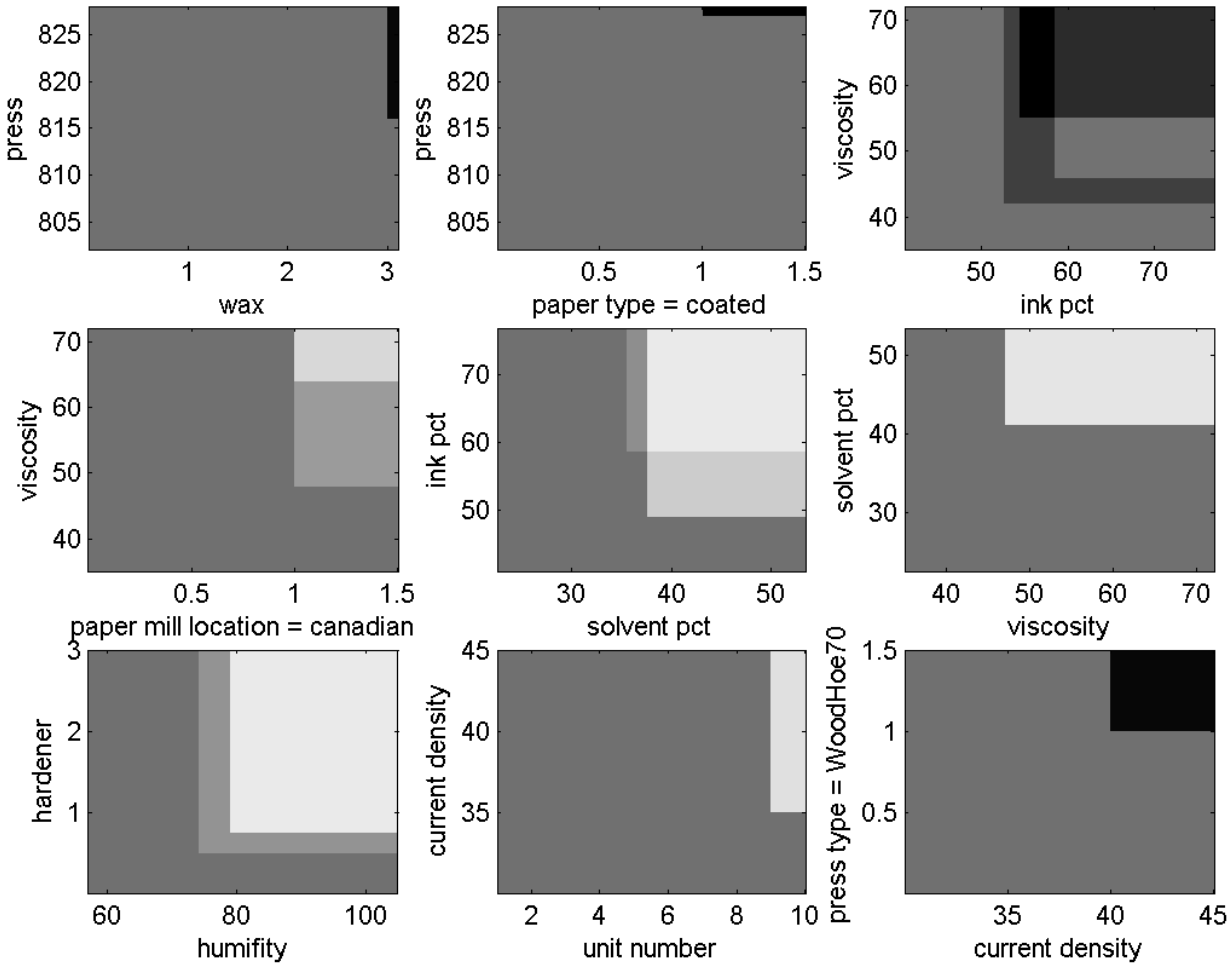

Figure 3: Role of the interaction in the score function. Database `bands`.

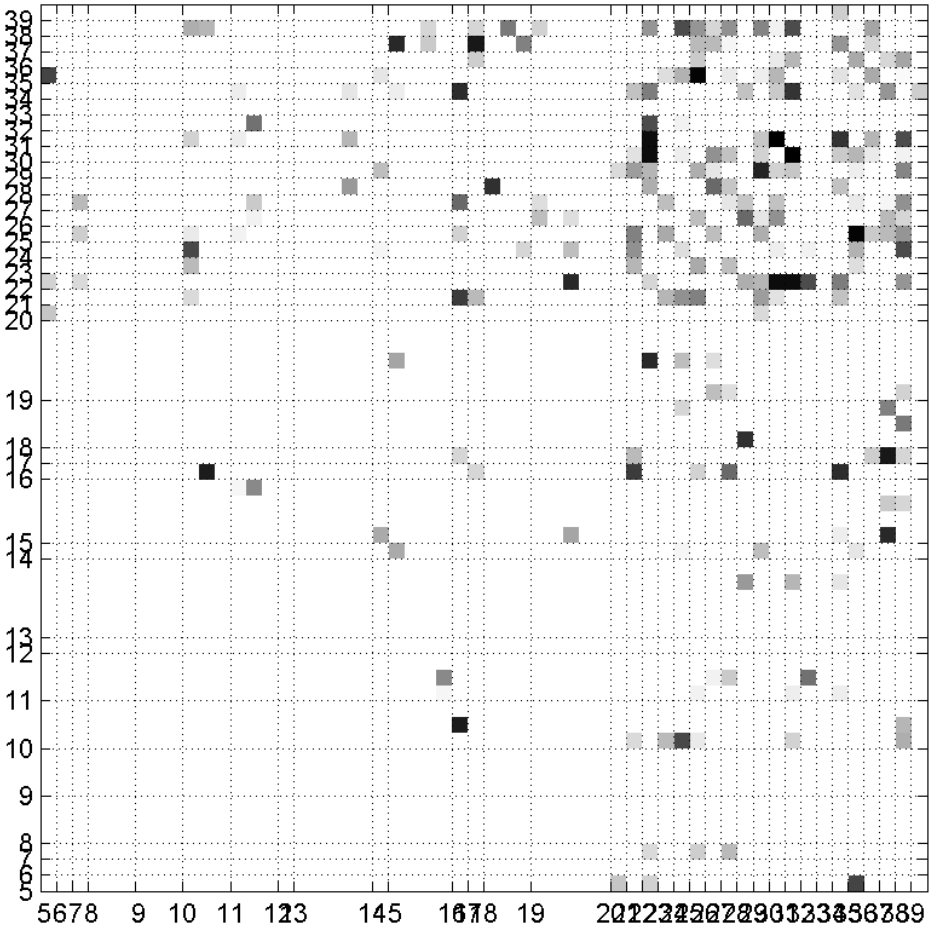

Figure 4: Interaction between pairs of variables. Database `bands`.

where the decision variables are the weight vector ω, the threshold value β and perturbations ξ^u associated with the misclassification of object $u \in I$. $\|\cdot\|$ denotes the L_1 norm, $N = \sharp(\mathcal{F})$ and C is a constant that trades off the margin in the correctly classified objects and the perturbations ξ^u. An appropriate value of C is usually chosen by crossvalidation techniques, see e.g. [9].

Whereas the distance between objects has usually been considered as the Euclidean between their predictor vectors, yielding the margin to be measured by the Euclidean norm as well, other norms such as the L_1 norm and the L_∞ norm have been considered, [10], and have been successfully applied, see for instance [1, 14, 17].

Contrary to the Euclidean case, in which a maximal margin hyperplane can be found by solving a quadratic program with linear constraints, in the L_1 norm case, used in this paper, an optimal solution can be found by solving a Linear Programming (LP) problem. In [12], empirical results show that 'in terms of separation performance, L_1, L_∞ and Euclidean norm-based SVM tend to be quite similar'. Moreover, polyhedral norms, as L_1 norm, contribute to sparsity in the classifier, yielding ω with many components equal to zero, see for instance [4]. The choice of L_1 norm also allows us to include constraints on the relative importance of the variables, as suggested for instance, in p.87 of [15].

Since we use the L_1 norm, Problem (8) can be formulated as the following LP problem,

$$\begin{array}{lll} \min & \sum_{\phi\in\mathcal{F}}(\omega_\phi^+ + \omega_\phi^-) + C\sum_{u\in I}\xi^u & \\ s.t.: & \sum_{\phi\in\mathcal{F}}(\omega_\phi^+ - \omega_\phi^-)c^u\phi(x^u) + \beta c^u + \xi^u \geq 1 & \forall u \in I \\ & \omega_\phi^+ \geq 0 & \phi \in \mathcal{F} \\ & \omega_\phi^- \geq 0 & \phi \in \mathcal{F} \\ & \xi^u \geq 0 & \forall u \in I \\ & \beta \in I\!R. & \end{array} \tag{9}$$

After finding the maximal margin hyperplane in the feature space defined by $\mathcal{F}$, the score function has the form described in (6).

For each feature ϕ, the absolute value $|\omega_\phi| = \omega_\phi^+ + \omega_\phi^-$ of its coefficient indicates the importance of that feature for the classification. Using basic Linear Programming theory, we see that the number of

features with non-zero coefficient is not larger than the number of objects in the database. However, this number can still be too large for interpretability of the classifier.

3.2 Column Generation

In [2], for the particular case in which the set of features is just $\hat{\mathcal{F}}$, the well-known Mathematical Programming tool called Column Generation is proposed to solve Problem (9). Since it has a high number of decision variables, instead of solving it directly, the Column Generation technique solves a series of reduced problems where decision variables, corresponding to features in the set $\hat{\mathcal{F}}$, are iteratively added as needed. For the general case where interactions are taken into account, the same approach can be applied, but a new algorithm is needed to generate features of degree greater than one, as it will be proposed in Section 3.3.

For $F \subset \mathcal{F}$, let *Master Problem* (9-F) be Problem (9) with the family of features F. We start with an initial set of features F. For example, we can get as an initial set of features, the features generated by the BSVM Column Generation Algorithm (CG-BSVM) proposed by the authors in [2]. Other choices for the initial set of features might be useful, for instance, randomly generate a set of features, or one feature $\phi_{\ell b}$ per variable ℓ, with b equal to the median of variable x_ℓ in the objects of I. Once the initial set is generated, we solve Problem (9-F). The next step is to check whether the current solution is optimal for Problem (9) or not, and, in the latter case, generate a new feature ϕ improving the objective value of the current solution. The generated feature is added to the family of features F, and then Problem (9-F) is solved again. This process is repeated until no other promising feature is found. A simple summary scheme of the column generation algorithm can be seen below.

CG-summary: Summary of the column generation algorithm

Step 0. Get an initial set of features F.

Step 1. Generate one (or various) promising feature(s) ϕ.

Step 2. If no new promising features are found, then stop: we have found a good solution of Problem (9). Otherwise, add it (them) to the set F, solve Problem (9-F), and go to Step 1.

In order to generate new features, the Column Generation technique uses the dual formulation of Problem (9),

$$\begin{array}{lr} \max & \sum_{u\in I}\lambda_u \\ s.t.: & -1 \le \sum_{u\in I}\lambda_u c^u \phi(x^u) \le 1 \quad \forall \phi \in \mathcal{F} \\ & \sum_{u\in I}\lambda_u c^u = 0 \\ & 0 \le \lambda_u \le C \quad \forall u \in I. \end{array} \tag{10}$$

The dual formulation of the Master Problem (9-F) only differs from this one in the first set of constraints, which should be attained $\forall \phi \in F$ instead of $\forall \phi \in \mathcal{F}$.

Let (ω^*, β^*) be an optimal solution of Master Problem (9-F), let $(\lambda_u^*)_{u\in I}$ be the values of the corresponding optimal dual solution, and let $\Gamma(\phi) = \sum_{u\in I}\lambda_u^* c^u \phi(x^u)$. If the optimal solution of the Master Problem (9-F) is also optimal for Problem (9), then, for every feature $\phi \in \mathcal{F}$ the constraints of the Dual Problem (10) will hold, i.e.

$$-1 \le \Gamma(\phi) \le 1, \quad \forall \phi \in \mathcal{F}. \tag{11}$$

If (ω^*, β^*) is not optimal for Problem (9), then the most violated constraint gives us information about which feature is promising and could be added to F, in such a way that adding such a feature to the set F would yield, at that iteration, the highest improvement of the objective function. In this sense, the most promising feature $\phi \in \mathcal{F}$ to be added is that which maximizes $|\Gamma(\phi)|$. Finding such a feature ϕ can be reduced to solving two optimization problems:

$$\max_{\phi\in\mathcal{F}} \Gamma(\phi), \tag{12}$$

$$\min_{\phi\in\mathcal{F}} \Gamma(\phi). \tag{13}$$

class	x_1	x_2
c^1	5	8
c^2	9	3
c^3	6	6
c^4	3	10

Table 4: Simple example of database.

3.3 Generation of features

For the particular case in which just features of degree one are considered, an exact algorithm to find the optimal values of (12) and (13) is proposed in [2], the so-called Algorithm 1-BSVM. In this algorithm, for each predictor variable ℓ, the possible cutoffs are sorted and the optimal value of (12) and (13) are found. This exact algorithm cannot be directly extended for features of degrees greater than one. For instance, when we want to generate a feature of degree 2, $\phi_{\ell_1,\ell_2,b_1,b_2}$, four parameters are to be determined: two predictor variables ℓ_1, ℓ_2, and two cutoffs b_1, b_2, one for each chosen predictor variable. If, as done in [2], we first consider the predictor variables ℓ_1, ℓ_2 are fixed, then it is not possible to sort the objects simultaneously according to two variables, so Algorithm 1-BSVM described in [2] does not apply. Now we present a simple example where a local search heuristic procedure is used to generate promising features.

Example 6 *In Table 4, a simple example with two predictor variables and four objects is presented, where we generate all the possible features. At a certain step of the column generation algorithm, let λ_u be the dual values of the optimal solution of the reduced problem. We face the problem of generating the most promising feature. For predictor variable* 1*, the possible cutoffs are* $5, 6$ *and* 9 *while the possible cutoffs for predictor variable* 2 *are* $6, 8$ *and* 10*. We also consider here the trivial cutoff* 3 *for each predictor variable, which leads to a feature $\phi_{\ell,3}(x^u) = 1$ for all objects $u \in I$. In Table 5, the values of Γ for different cutoffs for predictor variable* 1 *(columns) and different cutoffs for predictor variable* 2 *(rows) are shown.*

Since we are also considering the trivial cutoffs, we can find in Table 5 the features of degree 1*, represented in the first column and the first row of the table, along with the features of degree* 2*. Note that, when we move along the positions in the table, right-to-left and up-to-down movements lead to either*

$\lambda^1c^1+\lambda^2c^2+\lambda^3c^3+\lambda^4c^4$	$\lambda^1c^1+\lambda^2c^2+\lambda^3c^3$	$\lambda^2c^2+\lambda^3c^3$	λ^2c^2
$\lambda^1c^1 \quad +\lambda^3c^3+\lambda^4c^4$	$\lambda^1c^1 \quad +\lambda^3c^3$	λ^3c^3	-
$\lambda^1c^1 \quad +\lambda^4c^4$	λ^1c^1	-	-
λ^4c^4	-	-	-

Table 5: Generated features of degrees one and two.

Γ *gaining one term of the form* $\lambda_u c^u$ *or* Γ *remaining itself. This will be taken into account in order to implement heuristics to find a promising feature* ϕ *to be added to* F.

We now describe how to optimize Γ. Since the number of possible cutoffs is huge, we discard sing a global optimization procedure, and a local search will be used instead. To do this, we wil determine heuristically the cutoffs $b_1, b_2, \ldots, b_g$ associated with variables $\ell_1, \ell_2, \ldots, \ell_g$.

Suppose we know the value of Γ for certain feature of degree two, $\phi = \phi_{\ell_1,\ell_2,b_1,b_2}$. In Example 6, this corresponds to a position in Table 5. Once a predictor variable is fixed, for instance the predictor variable ℓ_1, the objects of I can be sorted increasingly by their values in that predictor variable. Let $u(i)$ the object in the i-th position. We want to change the current cutoff $b_1 = x_{\ell_1}^{u(i)}$ in order to create a different feature $\hat{\phi} = \phi_{\ell_1,\ell_2,\hat{b}_1,b_2}$. When changing b_1 to a different value $\hat{b}_1 = x_\ell^{u(j)}$, the values of ϕ change only in the objects $u(k)$ with $i < k \leq j$ if $i < j$ (and, respectively $j < k \leq i$ if $j < i$). Taking this into account, we know that, when moving backward, i.e. $j < i$, then the change in Γ is easily computed using that

$$\Gamma(\hat{\phi}) = \Gamma(\phi) - \sum_{k:\, \phi(x^{u(k)})=1,\, j<k\leq i} \lambda^{u(k)} c^{u(k)}.$$

When moving forward, i.e. $j > i$, then, in order to know if the term $\lambda^{u(k)}c^{u(k)}$ must be added, we need to check, for all objects $u(k)$ with $i < k \leq j$, whether $x_{\ell_1}^{u(k)}$ is greater than or equal to the cutoff $\hat{b}_1$. When dealing with features of degree greater than two, this last condition should be checked for all the other variables used in the feature, but the rest remains analogous.

All the above comments are taken into account to implement a local search heuristic, for features of degree up to g, that, in every step, moves forward or backward in the ordered set of possible cutoffs for a randomly chosen variable. The algorithm stops when T movements are performed without any

improvement in the value of Γ.

Algorithm 1-NBSVM: choosing g cutoffs.

Step 0. Initialization:

- $i_k \leftarrow 1$, $\forall k = 1, 2, \ldots, g$.
- $b_k \leftarrow x_{\ell_k}^{u(i_k)}$, $\forall k = 1, 2, \ldots, g$.
- `steps` $\leftarrow 0$ and max$\leftarrow 0$.

Step 1. Randomly choose a predictor variable $\ell_j \in \{\ell_1, \ell_2, \ldots, \ell_g\}$.

Step 2. Randomly choose a type of movement: `forward` or `backward`.

Step 3.
- If `forward`, then randomly choose $h \in \{i_k, i_k + 1, \ldots, \sharp(I)\}$.
- If `backward`, then randomly choose $h \in \{1, 2, \ldots, i_k\}$.

Step 4. Let $b = x_{\ell_j}^{u(h)}$, compute $\Gamma(\phi)$, for $\phi = \phi_{\ell_1,b_1} \ldots \phi_{\ell_{j-1},b_{j-1}} \cdot \phi_{\ell_j,b} \cdot \phi_{\ell_{j+1},b_{j+1}} \ldots \phi_{\ell_g,b_g}$.

Step 5. If $\Gamma(\phi) >$ `max` then $b^j \leftarrow b$, `max`$\leftarrow \Gamma(\phi)$ and `steps` $\leftarrow 0$. Otherwise, `steps` $\leftarrow$ `steps`+1.

Step 6. If `steps` <T, then go to Step 1. Otherwise, stop.

An analogous algorithm for minimizing Γ can be developed just changing Step 5 in the obvious way.

3.4 Implementation details

The column generation algorithm has been implemented as follows. First of all, the Algorithm CG-BSVM, described in [2], is run. Algorithm CG-BSVM obtains a solution of Problem (9) for the set $\hat{\mathcal{F}}$ of features of degree one, and, as a byproduct, it also provides us with a set of features F, that have been generated during its application. We take such a set of features F, as an initial set $F \subset \hat{\mathcal{F}} \subset \mathcal{F}$ in Step 0 of the scheme presented in (CG-summary). In a second step, features of degree up to g are generated. For a given set of g variables $\{\ell_1, \ell_1, \ldots, \ell_g\}$, Algorithm 1-NBSVM described in Section 3.3 provides us with

a local search heuristic to find the cutoffs $\{b_1, b_2, \ldots, b_g\}$, such that the feature $\phi = \phi_{\ell_1 b_1} \cdot \phi_{\ell_2 b_2} \cdot \ldots \cdot \phi_{\ell_g b_g}$ is promising to solve Problem (9).

In our implementation, g predictor variables are randomly selected. Other ways of selecting the g predictor variables might accelerate the optimization of Problem (9). We do not require the g predictor variables to be different, allowing in this way features of degree lower than g. For instance if the feature generated is $\phi = \phi_{4,0.5}\phi_{7,7.3}\phi_{7,8.9}$, it can be simplified to $\phi = \phi_{4,0.5}\phi_{7,8.9}$ which is a feature of degree two. For such set of g predictor variables, the local search heuristic described in Algorithm 1-NBSVM chooses their corresponding good cutoffs either by maximizing or minimizing Γ. We generate in this way Q features by maximizing Γ with Algorithm 1-NBSVM for Q different random choices of the set of predictor variables, and other Q features by minimizing Γ for other Q different random choices of the set of predictor variables. Among the generated features, those ϕ with $|\Gamma(\phi)| > 1$ are added to the set F and the LP problem (9-F) is solved. The whole algorithm of column generation stops when all of these $2Q$ generated features satisfy $|\Gamma(\phi)| < 1$. The current implementation of the column generation algorithm is as follows:

Column Generation Algorithm

Step 0. Run Algorithm CG-BSVM to build an initial set F with features of degree one.

Step 1. Repeat Q times:

- **Step 1.1.** Randomly choose a set of g predictor variables $\ell_1, \ell_2, \ldots, \ell_g$ and select $b_1, b_2, \ldots, b_g$ by maximizing Γ using **Algorithm 1-NBSVM**.
- **Step 1.2.** If $\Gamma(\phi) > 1$, then $F \leftarrow F \cup \{\phi\}$.

Step 2. Repeat Q times:

- **Step 2.1.** Randomly choose a set of g predictor variables $\ell_1, \ell_2, \ldots, \ell_g$ and use **Algorithm 1-NBSVM** for minimizing Γ, to choose $b_1, b_2, \ldots, b_g$.
- **Step 2.2.** If $\Gamma(\phi) < -1$, then $F \leftarrow F \cup \{\phi\}$.

Step 3. If F has not been modified in Steps 1 or 2, then stop: we have found a good solution of Problem (9). Otherwise, solve Problem (9-F) and go to Step 1.

4 Numerical results

First we are going to analyze the classification ability of the set of features proposed in the paper. With this aim, a series of numerical experiments have been performed using databases publicly available from the UCI Machine Learning Repository [11]. A summary of the characteristics of the databases used in the experiment is shown in Table 6. Five different databases were used, namely, the Cylinder Bands Database, called here `bands`; the Credit Screening Databases, called here `credit`; the Ionosphere Database, called here `ionosphere`; the Sonar Database, called here `sonar`; and the New Diagnostic Database, contained in the Wisconsin Breast Cancer Databases, called here `wdbc`.

For each database, the name of the file (as called in the database), the number of predictor variables (after codding nominal variables as explained in Section 2) p, and the total number of objects $\sharp(\Omega)$, are given in Table 6. In case of existence of missing values, as occurs in `bands` and `credit`, objects with missing values have been removed from the database.

name	filename	p	$\sharp(\Omega)$
`bands`	bands.data	56	365
`credit`	crx.data	43	666
`ionosphere`	ionosphere.data	34	351
`sonar`	sonar.all-data	60	208
`wdbc`	wdbc.data	30	569

Table 6: Information about the databases.

In order to compare the quality of the NBSVM classifier with the classification quality of other classifiers, we have tested the performance of two very different benchmark methods: classification trees, with and without pruning, and SVM with linear kernel, polynomial kernel for degrees between two and five, and radial basis function kernel. The averaged percentages of correctly classified objects in both the training sample (`tr`) and testing sample (`test`) are displayed in Table 7 for standard SVM and

different values of the parameter C, which trades off the margin and the perturbations in formulation (8). For classification trees, results are shown in Table 8. All results presented are obtained by 10-fold crossvalidation, e.g. [9]. CPLEX 8.1.0 was used as the LP solver.

Results of NBSVM are shown, for different values of C and g, in Tables 9-13, where the averaged percentages of correctly classified objects in training and testing samples are displayed along with the number of generated features (♯ f) with non-zero coefficient in the classifier and the number of predictor variable actually used by the classifier (♯ v). As NBSVM is an extension of BSVM, we have included the results of BSVM, which coincides with NBSVM for $g = 1$, in the first group of rows in Tables 9-13.

In order to interpret the obtained classifier, the number of features cannot be high. It might be useful to keep the number of features actually used by the classifier low. As in [2], it is also possible to reduce the number of features by using a wrapper approach that recursively deletes features. For instance, [5] proposes a procedure, successfully applied in standard linear SVM, where all the generated features with zero coefficient in the classifier and the feature whose non-zero coefficient has smallest absolute value are eliminated. Then, the coefficients are recomputed by the optimization of the LP Problem (9). This elimination procedure is repeated until the number of features is below a desired threshold.

It is a well-known fact in SVM that, if the parameter C is chosen too close to zero, one may obtain as optimal solution of Problem (9) a vector with $\omega = 0$, from which a trivial classifier assigning all objects to one class is obtained. This degenerate situation (indicated in Tables 9-13 as d.c.) is avoided by taking a bigger C.

The results show that NBSVM behaves comparable to the standard SVM technique. Indeed the best averaged percentage of correctly classified objects, for the best choice of C, of standard SVM is never worse than NBSVM's more than 0.86%.

Comparing NBSVM with its ancestor BSVM, we see that the consideration of interactions via the introduction of features of degree greater than one leads to an improvement in the classification performance, together with the added value of allowing us to measure interactions intensity, as illustrated in Section 2.

Moreover, comparing Table 8 with Tables 9-13, it turns out that, in terms of classification performance, NBSVM generally behaves considerably better than classification trees. For instance, taking the database `bands`, for any choice of C the prediction rate of NBSVM is nearly 5 points higher than Classification Trees'. Classification trees are widely used in applied fields as diverse as Medicine (diagnosis), Computer Science (data structures), Botany (classification), and Psychology (decision theory), mainly because they are easy to interpret. We claim that the column generation approach proposed in this paper maintains this property without losing the good classification ability of standard SVM.

If we want to keep low the number of features used, we can use the wrapped (instead of the crude) version of NBSVM. From our computational experience it seems that the wrapping slightly worsens the classification ability in most instances, though in some cases it deteriorates significantly. This is the case, for instance, of `sonar`. However, even the wrapped version behaves better or equal than Classification Trees (even 7 points above) for all values of the parameter C, as shown in Tables 8 and 12.

5 Conclusions and further research

In this paper an extension of BSVM for supervised classification, the NBSVM, has been proposed, where the classifier gives insightful knowledge about the way the predictor variables and the interactions between them influence the classification. Indeed, the nonlinearity behavior of the data and interactions between the different variables is modelled by the NBSVM classifier using simple queries, of type (2), and combinations of them, easily interpretable by practitioners, for instance by representations similar to Figure 1. Its classification behavior, which is between SVM and Classification Trees, makes NBSVM an interesting tool when a good classification ability is required, but interpretability of the results is an important issue.

Although in most instances the wrapping procedure leads to a very slight deterioration of the classification ability, in some cases the change is considerable. The design of more sophisticated wrapping strategies deserves further study.

Acknowledgments

This work has been partially supported by projects of MEC MTM2005-09362-C03-01, Spain, and FQM-329 of Junta de Andalucía, Spain.

References

[1] K. Bennet. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 307–326. MIT Press, 1999.

[2] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. A column generation approach for support vector machines. Technical report, Optimization Online, 2006. `http://www.optimization-online.org/DB_HTML/2006/04/1359.html`.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[4] G. Fung and O.L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[6] H. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[7] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

[8] R. Herbrich. *Learning Kernel Classifiers. Theory and Algorithms*. MIT Press, 2002.

[9] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1137–1143. Morgan Kaufmann, 1995.

[10] O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[11] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.

[12] J.P. Pedroso and N. Murata. Support vector machines with different norms: motivation, formulations and results. *Pattern Recognition Letters*, 22:1263–1272, 2001.

[13] D.K. Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nature Genetics Supplement*, 32:502–508, 2002.

[14] A. Smola, T.T. Friess, and B. Schölkopf. Semiparametric support vector and linear programming machines. In M.J. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 10*, pages 585–591. MIT Press, 1999.

[15] L.C. Thomas, D.B. Edelman, and J.N. Crook. *Credit scoring and its applications*. SIAM, 2002.

[16] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[17] J. Weston, A. Gammerman, M.O. Stitson, V. Vapnik, V. Vovk, and C. Watkins. Support vector density estimation. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 293–305. MIT Press, 1999.

bands												
	lineal		degree 2		degree 3		degree 4		degree 5		rbf	
C	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test
0.01	64.44	64.44	80.74	74.44	97.24	78.15	100.00	75.56	100.00	75.56	64.44	64.44
0.10	74.57	65.93	90.45	75.93	100.00	75.93	100.00	75.56	100.00	75.56	64.44	64.44
1	80.16	71.85	99.14	77.41	100.00	75.93	100.00	75.56	100.00	75.56	95.43	73.33
10	81.65	72.96	100.00	73.33	100.00	75.93	100.00	75.56	100.00	75.56	100.00	72.96
100	82.18	72.22	100.00	73.33	100.00	75.93	100.00	75.56	100.00	75.56	100.00	72.96
1000	82.35	72.59	100.00	73.33	100.00	75.93	100.00	75.56	100.00	75.56	100.00	72.96
credit												
	lineal		degree 2		degree 3		degree 4		degree 5		rbf	
C	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test
0.01	86.36	86.31	86.70	86.15	92.99	85.54	96.44	84.92	98.77	81.69	54.77	54.77
0.10	86.31	86.31	90.87	85.08	96.36	84.00	98.79	80.46	99.62	77.69	74.53	70.77
1	86.74	85.85	95.16	84.92	98.24	80.46	99.52	77.85	99.71	77.38	95.08	85.69
10	86.80	86.00	96.96	83.85	99.32	77.69	99.71	76.77	99.78	77.38	97.42	85.85
100	86.91	85.85	98.44	78.62	99.73	77.08	99.78	76.92	100.00	76.15	99.13	83.85
1000	87.09	85.54	99.62	78.15	99.78	76.31	100.00	75.38	100.00	76.15	99.71	82.77
ionosphere												
	lineal		degree 2		degree 3		degree 4		degree 5		rbf	
C	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test
0.01	66.25	65.71	91.52	88.86	97.71	91.14	99.43	90.29	100.00	86.57	64.00	64.00
0.10	89.21	87.71	95.87	90.57	99.37	90.57	100.00	86.29	100.00	86.57	94.76	93.14
1	91.84	87.71	98.29	90.29	100.00	87.43	100.00	86.29	100.00	86.57	98.00	93.71
10	94.03	88.29	99.52	90.00	100.00	87.71	100.00	86.29	100.00	86.57	99.49	94.00
100	95.21	87.71	100.00	87.43	100.00	87.71	100.00	86.29	100.00	86.57	100.00	94.29
1000	95.65	86.29	100.00	87.43	100.00	87.71	100.00	86.29	100.00	86.57	100.00	94.29
sonar												
	lineal		degree 2		degree 3		degree 4		degree 5		rbf	
C	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test
0.01	54.56	54.00	88.78	79.00	99.39	83.50	100.00	86.00	100.00	85.00	53.00	53.00
0.10	84.33	75.00	97.00	78.50	100.00	86.50	100.00	86.00	100.00	85.00	53.39	53.50
1	88.39	74.50	100.00	86.00	100.00	86.50	100.00	86.00	100.00	85.00	100.00	86.00
10	92.28	73.50	100.00	86.50	100.00	86.50	100.00	86.00	100.00	85.00	100.00	87.50
100	97.06	75.00	100.00	86.50	100.00	86.50	100.00	86.00	100.00	85.00	100.00	87.50
1000	100.00	73.50	100.00	86.50	100.00	86.50	100.00	86.00	100.00	85.00	100.00	87.50
wdbc												
	lineal		degree 2		degree 3		degree 4		degree 5		rbf	
C	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test
0.01	87.16	86.79	95.36	95.18	96.37	96.25	98.06	97.86	98.31	97.68	76.35	76.25
0.10	95.95	95.71	97.80	97.14	98.25	98.21	98.65	97.50	99.19	96.96	95.54	95.36
1	98.27	98.04	98.39	97.68	98.97	97.68	99.44	96.79	99.92	96.61	98.35	97.86
10	98.45	97.68	99.25	97.86	99.66	96.96	100.00	96.43	100.00	96.07	99.13	98.21
100	99.11	96.96	99.98	96.43	100.00	96.25	100.00	96.43	100.00	96.07	100.00	96.96
1000	99.50	96.43	100.00	96.96	100.00	96.25	100.00	96.43	100.00	96.07	100.00	96.61

Table 7: Results for SVM.

	bands		credit		ionosphere		sonar		wdbc	
	% tr	% test	% tr	% test	% tr	% test	% tr	% test	% tr	% test
Pruned Tree	74.12	64.81	86.31	86.31	91.17	89.43	82.56	71.50	96.71	94.46
Crude Tree	92.43	66.67	95.15	81.69	98.03	87.71	97.56	77.00	99.31	93.75

Table 8: Results for Clasification Trees.

		Crude				Wrapped	
g	C	% `tr`	% `test`	♯ `f`	♯ `v`	% `tr`	% `test`
1	0.01		d.c.			d.c.	
1	0.1		d.c.			d.c.	
1	1	98.85	73.33	121.1	28.0	92.14	72.22
1	10	100.00	72.59	128.8	28.3	94.81	66.30
1	100	100.00	72.22	128.5	28.3	95.47	66.30
1	1000	100.00	72.59	128.4	28.4	95.14	66.67
2	0.01		d.c.			d.c.	
2	0.1		d.c.			d.c.	
2	1	100.00	72.59	132.1	33.8	99.34	72.22
2	10	100.00	77.78	131.5	34.9	100.00	72.22
2	100	100.00	75.93	133.2	34.7	100.00	73.33
2	1000	100.00	75.56	134.2	34.6	100.00	75.56
3	0.01		d.c.			d.c.	
3	0.1	88.23	73.33	39.0	28.0	87.41	72.96
3	1	100.00	74.07	140.9	35.6	99.92	76.30
3	10	100.00	76.30	143.4	35.0	100.00	70.74
3	100	100.00	77.41	143.2	34.8	100.00	74.07
3	1000	100.00	78.15	141.6	35.6	100.00	72.96
4	0.01		d.c.			d.c.	
4	0.1	89.30	72.59	43.6	29.7	89.14	71.85
4	1	100.00	76.30	142.4	35.8	99.84	71.48
4	10	100.00	76.30	148.4	35.3	100.00	74.81
4	100	100.00	72.59	145.0	35.3	100.00	71.48
4	1000	100.00	78.52	147.4	35.0	100.00	73.33
5	0.01		d.c.			d.c.	
5	0.1	90.33	71.11	55.7	31.0	89.67	71.11
5	1	100.00	77.04	148.0	35.2	100.00	75.19
5	10	100.00	75.56	146.0	35.2	100.00	75.19
5	100	100.00	77.78	144.3	35.0	100.00	74.07
5	1000	100.00	77.78	148.2	35.3	100.00	73.33

Table 9: Classification behavior. Database `bands`.

		Crude				Wrapped	
g	C	% `tr`	% `test`	♯ `f`	♯ `v`	% `tr`	% `test`
1	0.01	86.31	86.31	1.0	1.0	86.31	86.31
1	0.1	86.31	86.31	1.0	1.0	86.31	86.31
1	1	95.71	82.92	138.2	21.7	91.93	84.00
1	10	100.00	80.00	199.5	24.8	89.50	80.92
1	100	100.00	79.69	200.3	24.7	90.09	82.00
1	1000	100.00	80.31	200.5	24.8	91.42	80.77
2	0.01	86.31	86.31	1.0	1.0	86.31	86.31
2	0.1	86.31	86.31	1.0	1.0	86.31	86.31
2	1	99.56	83.85	169.2	30.7	95.35	84.15
2	10	100.00	82.92	180.1	31.2	95.06	83.38
2	100	100.00	82.15	182.7	30.8	94.97	83.85
2	1000	100.00	81.69	181.8	30.3	95.11	81.23
3	0.01	86.31	86.31	1.0	1.0	86.31	86.31
3	0.1	86.32	86.00	1.0	1.1	86.32	86.00
3	1	99.93	83.54	188.4	29.0	95.86	83.54
3	10	100.00	85.85	193.7	29.3	96.48	81.85
3	100	100.00	83.85	192.6	29.5	96.92	82.31
3	1000	100.00	84.15	190.3	29.1	96.44	80.92
4	0.01	86.31	86.31	1.0	1.0	86.31	86.31
4	0.1	86.32	86.00	1.0	1.1	86.32	86.00
4	1	99.97	84.31	204.2	29.3	96.50	82.92
4	10	100.00	84.31	206.1	29.4	97.01	81.23
4	100	100.00	85.69	199.9	28.6	97.38	81.69
4	1000	100.00	85.54	198.5	28.5	97.74	81.85
5	0.01	86.31	86.31	1.0	1.0	86.31	86.31
5	0.1	86.32	86.00	1.0	1.1	86.32	86.00
5	1	100.00	84.92	209.1	28.4	96.27	83.38
5	10	100.00	85.38	213.0	27.8	97.44	80.46
5	100	100.00	83.85	207.8	28.3	97.45	80.46
5	1000	100.00	85.08	209.5	28.2	98.10	82.15

Table 10: Classification behavior. Database `credit`.

		Crude				Wrapped	
g	C	% `tr`	% `test`	♯ `f`	♯ `v`	% `tr`	% `test`
1	0.01		d.c.			d.c.	
1	0.1	91.17	90.57	2.0	2.0	91.17	90.57
1	0.1	100.00	90.57	92.7	31.1	100.00	90.00
1	10	100.00	90.57	93.1	31.2	100.00	89.43
1	100	100.00	90.57	93.1	31.2	100.00	89.71
1	1000	100.00	90.57	93.4	31.1	100.00	89.43
2	0.01		d.c.			d.c.	
2	0.1	94.10	90.57	11.2	10.1	94.10	90.57
2	0.1	100.00	92.57	105.6	32.4	100.00	91.14
2	10	100.00	92.57	107.3	31.9	100.00	91.71
2	100	100.00	92.57	109.5	32.0	100.00	91.71
2	1000	100.00	92.00	108.1	32.1	100.00	91.71
3	0.01		d.c.			d.c.	
3	0.1	94.89	90.86	16.5	13.3	94.89	90.57
3	0.1	100.00	92.29	101.2	32.6	100.00	91.43
3	10	100.00	92.29	104.4	32.7	100.00	92.29
3	100	100.00	92.86	103.0	32.8	100.00	91.71
3	1000	100.00	92.00	104.5	32.6	100.00	90.29
4	0.01		d.c.			d.c.	
4	0.1	95.05	91.71	20.2	16.2	95.05	92.29
4	0.1	100.00	92.57	104.1	33.0	100.00	92.57
4	10	100.00	92.00	99.5	33.0	100.00	91.43
4	100	100.00	92.57	102.5	33.0	100.00	92.57
4	1000	100.00	92.29	102.4	33.0	100.00	93.14
5	0.01		d.c.			d.c.	
5	0.1	95.59	92.57	35.1	24.8	95.62	92.57
5	0.1	100.00	93.43	100.8	32.8	100.00	91.71
5	10	100.00	92.29	101.5	33.0	100.00	92.57
5	100	100.00	93.14	98.7	33.0	100.00	92.57
5	1000	100.00	93.43	101.9	33.0	100.00	93.14

Table 11: Classification behavior. Database `ionosphere`.

		Crude				Wrapped	
g	C	% `tr`	% `test`	♯ `f`	♯ `v`	% `tr`	% `test`
1	0.01		d.c.			d.c.	
1	0.1	91.83	75.00	39.2	25.9	91.94	76.50
1	0.1	100.00	80.50	97.5	47.8	100.00	77.50
1	10	100.00	80.00	97.4	47.8	100.00	77.00
1	100	100.00	80.50	97.5	47.8	100.00	77.00
1	1000	100.00	80.50	97.5	47.8	100.00	77.00
2	0.01		d.c.			d.c.	
2	0.1	99.06	80.50	63.7	48.4	98.44	81.50
2	0.1	100.00	85.00	109.0	55.6	100.00	84.00
2	10	100.00	85.00	107.7	56.7	100.00	84.00
2	100	100.00	86.00	107.6	56.3	100.00	82.50
2	1000	100.00	86.00	107.6	56.3	100.00	82.50
3	0.01		d.c.			d.c.	
3	0.1	99.50	81.00	78.5	55.5	99.22	79.00
3	0.1	100.00	87.00	116.6	59.0	100.00	83.50
3	10	100.00	83.00	114.7	59.5	100.00	79.00
3	100	100.00	84.50	113.1	59.4	100.00	83.50
3	1000	100.00	84.50	113.1	59.4	100.00	83.50
4	0.01		d.c.			d.c.	
4	0.1	99.83	82.00	97.0	59.1	99.61	78.50
4	0.1	100.00	84.00	119.7	59.8	100.00	81.50
4	10	100.00	83.00	119.0	59.7	100.00	80.00
4	100	100.00	83.00	118.9	59.8	100.00	80.00
4	1000	100.00	83.00	118.9	59.8	100.00	80.00
5	0.01		d.c.			d.c.	
5	0.1	99.67	80.50	100.4	59.9	99.61	80.00
5	0.1	100.00	82.00	122.9	60.0	100.00	79.50
5	10	100.00	84.00	123.9	59.8	100.00	79.00
5	100	100.00	79.00	120.0	59.7	100.00	77.00
5	1000	100.00	79.00	120.0	59.7	100.00	77.00

Table 12: Classification behavior. Database `sonar`.

		Crude				Wrapped	
g	C	% `tr`	% `test`	♯ `f`	♯ `v`	% `tr`	% `test`
1	0.01	92.60	90.54	1.0	1.0	92.60	90.54
1	0.1	97.74	96.07	21.1	9.0	97.74	95.89
1	0.1	100.00	95.71	68.4	24.8	100.00	95.71
1	10	100.00	96.25	68.2	25.0	100.00	96.07
1	100	100.00	95.89	67.6	25.0	100.00	96.07
1	1000	100.00	96.07	68.0	25.0	100.00	96.25
2	0.01	93.49	91.07	1.0	1.8	93.49	91.07
2	0.1	98.95	95.89	41.0	23.3	99.01	95.89
2	0.1	100.00	96.79	87.4	29.7	100.00	96.79
2	10	100.00	97.14	90.0	29.4	100.00	97.14
2	100	100.00	97.32	88.0	29.5	100.00	96.79
2	1000	100.00	96.61	89.9	29.5	100.00	96.61
3	0.01	93.67	91.07	1.0	2.6	93.67	91.07
3	0.1	98.95	95.00	42.6	25.5	98.95	95.36
3	0.1	100.00	96.61	103.0	29.7	100.00	95.89
3	10	100.00	97.14	99.9	30.0	100.00	96.25
3	100	100.00	97.50	103.3	30.0	100.00	96.61
3	1000	100.00	96.79	101.5	29.9	100.00	96.25
4	0.01	94.15	93.04	1.0	3.4	94.15	93.04
4	0.1	99.19	95.36	54.8	29.5	99.21	94.82
4	0.1	100.00	97.14	110.1	30.0	100.00	96.61
4	10	100.00	96.07	112.3	30.0	100.00	95.54
4	100	100.00	97.14	108.2	30.0	100.00	96.61
4	1000	100.00	97.50	112.1	30.0	100.00	96.61
5	0.01	94.15	91.79	1.0	4.0	94.15	91.79
5	0.1	99.23	95.54	54.2	27.8	99.19	95.89
5	0.1	100.00	95.89	114.4	30.0	100.00	95.18
5	10	100.00	96.25	110.8	30.0	100.00	96.43
5	100	100.00	96.43	113.3	30.0	100.00	96.07
5	1000	100.00	96.61	115.1	30.0	100.00	95.89

Table 13: Classification behavior. Database `wdbc`.