Mayhua-López, E., Gómez-Verdejo, V. & Figueiras-Vidal, A. R. (2015). A new boosting design of Support Vector Machine classifiers. *Information Fusion, 25*, 63–71.

DOI: 10.1016/j.inffus.2014.10.005

A New Boosting Design of Support Vector Machine Classifiers

Efraín Mayhua-López[a], Vanessa Gómez-Verdejo[b], Aníbal R. Figueiras-Vidal[b]

[a] *Universidad Católica San Pablo, Arequipa, Perú.*
[b] *Department of Signal Theory and Communications, Universidad Carlos III de Madrid, 28911 Madrid, Spain.*

Abstract

Boosting algorithms pay attention to the particular structure of the training data when learning, by means of iteratively emphasizing the importance of the training samples according to their difficulty for being correctly classified. If common kernel Support Vector Machines (SVMs) are used as basic learners to construct a Real AdaBoost ensemble, the resulting ensemble can be easily compacted into a monolithic architecture by simply combining the weights that correspond to the same kernels when they appear in different learners, avoiding to increase the operation computational effort for the above potential advantage. This way, the performance advantage that boosting provides can be obtained for monolithic SVMs, i.e., without paying in classification computational effort because many learners are needed. However, SVMs are both stable and strong, and their use for boosting requires to unstabilize and to weaken them. Yet previous attempts in this direction show a moderate success.

In this paper, we propose a combination of a new and appropriately designed subsampling process and an SVM algorithm which permits sparsity

control to solve the difficulties in boosting SVMs for obtaining improved performance designs. Experimental results support the effectiveness of the approach, not only in performance, but also in compactness of the resulting classifiers, as well as that combining both design ideas is needed to arrive to these advantageous designs.
Keywords: Classification, Real AdaBoost, subsampling, Support Vector Machines, Linear Programming, ensemble classifiers, boosting.

---

1. Introduction

1.1. Background

The concept of Maximal Margin (MM) for classification purposes was introduced by Vapnik in his works on statistical learning [1–3]. The application of the Representer Theorem [4] and the Lagrangian formulations of its overlapping class versions originated the Support Vector Machines (SVMs) [5–10], an appreciated family of algorithms to optimizes a unique extremum functional which much deserve the general interest and the wide use they have found, even in other research areas, such as some regression forms in Digital Signal Processing (DSP) [11–14].

The basic standard form of the optimization for designing a binary SVM classifier is:

$$\begin{aligned} \min_{\mathbf{w},b,\{\xi^{(l)}\}} \quad & \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{l=1}^{L}\xi^{(l)}\right] \\ \text{s.t.} \quad & y^{(l)}\left[\langle \mathbf{w}, \Phi(\mathbf{x}^{(l)})\rangle + b\right] \geq 1 - \xi^{(l)}, \quad l = 1,\ldots,L \\ & \xi^{(l)} \geq 0, \qquad\qquad\qquad\qquad\quad l = 1,\ldots,L \end{aligned} \tag{1}$$

where $\{x^{(l)}, y^{(l)}\}$ are the labeled training examples, $\Phi(\cdot)$ is some nonlinear transformation (which is absorbed by means of the kernel trick), $\{\xi^{(l)}\}$ are (non-negative) slack variables that serve to penalize out-of-margin sample locations, and, finally, $C > 0$ is a regularization parameter that controls the trade-off between the margin inverse measure $\|\mathbf{w}\|^2$ and the overall penalization. An alternative vision [15] considers that (1) minimizes the sum of the slack variables plus $\frac{\|\mathbf{w}\|^2}{2}$, which plays the role of a conventional regularization term. The optimization is solved by applying Lagrange multipliers and introducing the dual form, a simple problem which can be solved by Quadratic Programming (QP) routines.

The above and many alternative forms, such as the quadratic penalization [16], the ν-SVM [5] and the (dual) Linear Programming SVM (LPSVM) [3, 17–20], provide sparse (i.e., excluding some of all the kernels that can appear in the solution) and high performance solutions as linear combinations of selected kernels $K(\cdot, \mathbf{x}^{(l)})$. There are other forms that propose more general functional optimizations [21], requiring to apply other search algorithms. Yet in all the cases the functionals to be optimized are "a priori" selected according to the general characteristics of the database under analysis and the interests of the designer, and this means that the intrinsic structure of the training dataset is not taken into account –as it occurs for most conventional training algorithms.

There are other machine learning processes that pay attention to the structure of the databases, such as emphasized soft target methods [22] or negative correlation learning [23, 24]. Among them, boosting algorithms emerge as a fundamental concept to construct classifier ensembles. Boosting

ensembles are built step–by–step, and each new learner is trained to minimize an overall error in which training examples receive a weight which is selected with the value of a exponential marging measure of the corresponding classification output of the ensemble which has bean built until adding this learner, i.e., examples that are difficult for being correctly classified receive more attention. So, the intrinsic characteristics of the problem are considered for designing the ensemble, since training examples are emphasized according to the difficulty of classifying them in a correct manner. On the other hand, the outputs of the learners are linearly combined. Note that this means that, if SVMs are used as learners, each kernel of each learner contributes to the overall output with its output value multiplied by a number (a weight parameter), and the overall output simply adds these contributions. After the pioneer AdaBoost (for binary output learners) [25] and Real AdaBoost (for continuous output learners) (RAB) [26] forms, a huge quantity of modifications and extensions have been proposed [27].

As for other ensembles, learner diversity (i.e., differences among their outputs in response to the same input) is necessary for the success of boosting algorithms. This diversity comes from the emphasis effect on unstable architectures, those sensitive to moderate changes in the training set. A key aspect of these architectures is the use of weak learners, i.e., units with moderate classification capabilities. This seems to be the origin of a relevant and somewhat surprising characteristic of boosting ensembles, their resistence to overfitting [28–30]. It is true that overfitting can appear under some difficult conditions, such as a high level of noise or if many outliers are included in the training set [31–33]. However, there are several modified algorithms that

reduce the corresponding negative effects [34–41].

1.2. Boosting SVM learners

Since the output of a boosting ensemble is a linear combination of its learners' outputs and the output of each SVM learner is a linear combination of its kernels' outputs, the overall output can be reduced to a linear combination of kernels' outputs, i.e., to a monolithic SVM structure. This is a promising avenue to design monolithic SVM classifiers with improved performance, which comes from the beneficial effects of the boosting emphasis process. But SVM classifiers are essentially stable –they are linear in the reduced kernel Hilbert space–, and they are also strong, and these facts provoke that a direct use of SVM as RAB learners does not improve the classification capabilities of the block designs, and in many cases these capabilities are even reduced [42]. Boosting designs require unstable and weak enough learners to obtain advantages of the step–by–step emphasis mechanism; if learners are stable, diversity do not appears, and if learners are strong, the process finishes in few steps. And to weaken SVM classification units by intentionally selecting "ad-hoc" values for the regularization parameters does not constitute a reasonable alternative, because the resulting designs tend to keep the same kernels (the same Support Vectors), and this reduces the necessary diversity.

On the one hand, this has conducted the recent research for designing boosting ensembles having local approximation capabilities along other directions, such as the indirect way [43], or including local gates in the aggregation step of global approximation learners [44], or even MM designs for other local-global big monolithic architectures that come from modifying other

ensembles [45, 46]. The resulting ensembles have excellent performances, but they are computationally demanding because learners remain separate [43, 44] or the number of elements becomes much higher than the number of training samples [45, 46].

On the other hand, several modes of unstabilizing and weakening SVM learners have been proposed. There are two main design approaches for it. The first family of these designs introduces diversity by varying the kernel characteristics along the growth of the ensemble [47–50]. In particular, the Diverse AdaBoost SVM (D-ABSVM) [50], which employs the traditional Gaussian kernels, additionally demands that each new learner presents some diversity with respect to the previous ones to be added to the ensemble. Obviously, this approach does not allow to reduce the ensemble to a monolithic equivalent. The second family of elaborated SVM boosting ensembles applies subsampling to obtain adequate SVM learners [51, 52]. One of them, the AdaBoost Weakness SV (AB-WSV) [52] also uses ν-SVM designs [5] for gaining an additional control of the characteristics of the base classifiers.

These two kinds of procedures to unstabilize and to weaken SVM learners showed a moderate success and their performance evaluation is not conclusive about their real advantage. The reasons seem to be the following. The subsampling approaches are prone to suffer the effects of frequently including the most missclassified samples in the different subsampled training sets, presenting overfitting problems. With respect to the algorithms that vary the kernel width, they can select outliers as SVs for large values of its width, and noisy samples if the width is small.

Therefore, it is clear that, to exploit the potential performance advantage

of SVM boosting designs, much care must be paid to the auxiliary mechanisms that are needed to introduce learner diversity and weakness. In this paper, we propose a combination of two components to deal with this challenge. First, a sophisticated structured subsampling mechanism which avoids the possibility of repeatedly selecting difficult samples, thus keeping diversity while weakening the learners, unlike other subsampling mechanism that have been used to force diversity among ensemble units [53, 54]. Second, a sparsity inducing SVM training algorithm, LPSVM [3, 17–20], is reformulated using a row subsampled kernel matrix to integrate the selected samples as kernels while training with all available samples. LPSVM does not create difficulties with weakening the learners and, at the same time, dual domain LP helps to control the number of kernels that are included in the SVM solution, contributing to the obtention of compact equivalent monolithic final architectures.

We want to emphasize that we have selected the above two components of our SVM boosting ensemble design procedure to effectively deal with the serious problems that appear when both diversity and weakness are required for SVM boosting learners, and it is their combination what permits the remarkable performance and compactness of the resulting designs, as the experiments we present in this paper clearly indicate.

The rest of the article is organized as follows. A brief look at RAB algorithm is included in Section 2. The proposed subsampled LPSVM formulation is described in Section 3. Section 4 shows the performance results of our approach in a number of benchmark problems, as well as the degradation that excluding one of the design components induces. The same section

discusses the effects of sampling intensity and the computational effort. We close the paper with our main conclusions and some suggestions for further research in Section 5.

2. A brief summary of Real AdaBoost

For the sake of clarity, we will restrict our work to binary problems. Multiclass problems can be addressed in an analogous manner by using multiclass monolithic designs of SVMs [55, 56] and multiclass versions of boosting [57] that are compatible with our formulation. To employ multiple binary formulations (one vs. one, one vs. rest, or Error Correcting Output codes [58, 59]) is an alternative possibility.

Let $\{\mathbf{x}^{(l)}, y^{(l)}\}_{l=1}^{L}$ be the training examples, where $\mathbf{x}^{(l)} \in \mathcal{R}^d$ is a d-dimensional input and $y^{(l)} \in \{-1, 1\}$ is its corresponding label. Considering the set of base classifiers $\{f_t(\mathbf{x})\}_{t=1}^{T}$, with $f_t(\mathbf{x}) \in [-1, 1]$, RAB constructs an ensemble classifier by a linear combination of these base learners as:

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}), \tag{2}$$

where $\alpha_t \geq 0$ is the weight assigned to the output of the t-th base classifier. The ensemble decision is made according to the sign of $F_T(\mathbf{x})$.

To train each base learner, the RAB algorithm makes learning to pay attention to the patterns according to an emphasis function over training samples, $D_t(l)$. The first learner considers equal weights for all data, $D_1(l) = 1/L \ \forall l$, and, iteratively, these weights are updated in such a way that the emphasis value assigned to high erroneous samples is increased, whereas it

is decreased for low error examples. Specifically, the weights are updated in accordance to the emphasis function

$$D_{t+1}(l) = \frac{D_t(l)\exp[-\alpha_t f_t(\mathbf{x}^{(l)})y^{(l)}]}{Z_t}, \tag{3}$$

where Z_t is a normalization factor to ensure $\sum_{l=1}^{L} D_{t+1}(l) = 1$.

The combination parameter associated to the output of the t-th base classifier is calculated by minimizing a bound of the exponential margin cost, obtaining

$$\alpha_t = \frac{1}{2}\ln\frac{1+r_t}{1-r_t}, \tag{4}$$

r_t being the edge of the base classifier, whose expression is

$$r_t = \sum_{l=1}^{L} D_t(l) f_t(\mathbf{x}^{(l)})y^{(l)}. \tag{5}$$

[26, 27] provide further details.

3. Training SVMs as RAB learners

3.1. Linear Programming SVM

The LPSVM [3, 17–20] is a modified version of the SVM formulation where the $L2$ penalty over the solution vector $\mathbf{w}$ is replaced by a $L1$ norm over its dual variables. To do so, we consider that the dual form solution which appears using the Representer Theorem [4]

$$\mathbf{w} = \sum_{l=1}^{L} a^{(l)}\boldsymbol{\phi}(\mathbf{x}^{(l)}), \tag{6}$$

where $\boldsymbol{\phi}(\cdot)$ is the corresponding mapping function $\Re^d \rightarrow RKHS$, has the form

$$o\left(\mathbf{x}\right) = \sum_{l'=1}^{L} a^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}) + b. \tag{7}$$

where K is the kernel $\boldsymbol{\phi}(\mathbf{x}^{(l)})^{\top}\boldsymbol{\phi}(\mathbf{x})$, and we minimize the sum of the corresponding slack variables and a regularization term which is the $L1$ norm of the variables $\left\{a^{(l')}\right\}$, arriving to the LPSVM formulation

$$\begin{array}{ll} \min\limits_{\mathbf{a},b,\xi} & \left\{ \left\|\mathbf{a}\right\|_1 + C\sum\limits_{l=1}^{L} \xi^{(l)} \right\} \\ \text{s.t.} & y^{(l)} \left[\sum\limits_{l'=1}^{L} a^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b \right] \geq 1 - \xi^{(l)}, \quad \forall l, \\ & \xi^{(l)} \geq 0, \qquad\qquad\qquad\qquad\qquad\qquad\quad \forall l. \end{array} \tag{8}$$

From this optimization problem, we obtain the optimum values of the parameters $\mathbf{a}$ and b and, then, compute the LPSVM output for any pattern according to

$$f\left(\mathbf{x}\right) = \sum_{l'=1}^{L} a^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}) + b. \tag{9}$$

It is important to note that, here, the role of parameter C has changed, being now a trade-off between the sum of slack variables and the level of sparsity in $\mathbf{a}$. Thus, it provides an easy mechanism to control the complexity of discriminant function (9).

3.2. The proposed subsampling procedure

As previously mentioned, to adopt an appropriate subsample procedure is a critical aspect in order to obtain high performance RAB designs when using SVM base learners.

According to [60], when the emphasis criterion is included in the subsampling procedure, there are three commonly used strategies: (1) Trimming, which selects only samples with a high emphasis weight; (2) Unique Uniform Sampling (UUS), where all samples have equal probability to be selected; and (3) Weighted Sampling (WS), which assigns each sample a different probability of being selected (depending on its emphasis weight), allowing replacement process.

Considering the emphasis effect during the subsampling process, trimming and WS are clearly inappropriate for our purposes, because samples that are far from the classification border will dominate. With respect to UUS, it would guarantee diversity, but not a compact final machine; besides, forgetting emphasized samples does not seem reasonable. But a combination of WS and UUS appears as an appropriate strategy: It will take into account compactness and, simultaneously, it can provide heterogeneous sample subsets including samples that are easy for labeling correctly, and others that are difficult to assign to the right class. Consequently, we propose the following new subsampling procedure.

Let L' be the number of samples to be selected. At each training epoch, samples are ordered according to their emphasis, and L' groups of the same (or similar) number of samples are created according to that order. For the next training epoch, one sample of each group is selected with equal probability. Obviously, the resulting subset of examples is an appropriate representation of the problem to be solved, because both correctly classified and misclassified samples are included.

3.3. Training subsampled LPSVM base learners

Once we apply the above subsampling procedure, the set of samples that are candidates to become SVs is accordingly indexed by $\mathcal{I}^{L'}$, where L' ($< L$) is the number of samples in the subset. The subsampled LPSVM (SLPSVM) is trained by solving the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{a},b,\xi} \quad \left\{ \sum_{l' \in \mathcal{I}_t^{L'}} \left| a^{(l')} \right| + C \sum_{l=1}^{L} D_t(l) \xi^{(l)} \right\} \\ &\text{s.t. :} \\ &y^{(l)} \left[\sum_{l' \in \mathcal{I}_t^{L'}} a^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b \right] \geq 1 - \xi^{(l)}, \quad \forall l, \\ &\xi^{(l)} \geq 0, \qquad \forall l. \end{aligned} \tag{10}$$

We have included the emphasis term $D_t(l)$ as a factor of the slack variable $\xi^{(l)}$, and $\mathcal{I}_t^{L'}$ is the set of subsampled examples in round t.

Note that the vector of dual variables, $\mathbf{a}$, can only have components in the subset indexed by $\mathcal{I}^{L'}$ (not all the data are allowed to be SVs). However, the classification constraints are evaluated for all the training data, i.e., optimization problem (10) tries to correctly classify all the data. As we will check in the experimental section, the improved performance of the proposed SLPSVM boosting ensemble, compared to other attempts of using SVM learners with subsampled data sets [51, 52], relies on this new subsampling procedure.

Finally, there are two issues to be clarified from a practical point of view. First, the presence of absolute values of $a_t^{l'}$ in (10) precludes the use of standard toolboxes. Fortunately, this drawback can be easily over-

come by redefining dual variables as $a_t^{(l')} = a_{t_+}^{(l')} - a_{t_-}^{(l')}$, with $a_{t_+}^{(l')}, a_{t_-}^{(l')} \geq 0$. Thus, the 1-norm over the dual variables can be expressed as $\sum_{l' \in \mathcal{I}_t^{L'}} |a_t^{(l')}| = \sum_{l' \in \mathcal{I}_t^{L'}} (a_{t_+}^{(l')} + a_{t_-}^{(l')})$, because the optimization forces that at least one of these terms, $a_{t_+}^{(l')}$ or $a_{t_-}^{(l')}$, is zero. Then, (10) can be converted to a linear programming (LP) problem:

$$\begin{aligned}
&\min_{\mathbf{a}_t, b_t, \xi} \quad \left\{ \sum_{l' \in \mathcal{I}_t^{L'}} \left(a_{t_+}^{(l')} + a_{t_-}^{(l')} \right) + C \sum_{l=1}^{L} D_t(l) \xi^{(l)} \right\} \\
&\text{s.t. :} \\
&y^{(l)} \left[\sum_{l'} \left(a_{t_+}^{(l')} - a_{t_-}^{(l')} \right) K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b_t \right] \geq 1 - \xi^{(l)}, \quad \forall l, \\
&\xi^{(l)} \geq 0, \quad \forall l, \\
&a_{t_+}^{(l')}, \quad a_{t_-}^{(l')} \geq 0, \quad l' \in \mathcal{I}_t^{L'},
\end{aligned} \tag{11}$$

and it can be solved by any LP toolbox. After computing the optimal values of $\{a_{t_+}^{(l')}\}_{l'=1}^{L'}$, $\{a_{t_-}^{(l')}\}_{l'=1}^{L'}$, b_t, the output of the t-th SLPSVM learner is given by

$$f_t(\mathbf{x}) = \sum_{l' \in \mathcal{I}_t^{L'}} \left(a_{t_+}^{(l')} - a_{t_-}^{(l')} \right) K(\mathbf{x}^{(l')}, \mathbf{x}) + b_t. \tag{12}$$

The output range (over training samples) of the SLPSVM has to be limited to the interval $[-1, 1]$ in order to allow adding it to the RAB ensemble. This can be easily achieved by dividing it by the maximum absolute value over the training samples

$$\tilde{f}_t(\mathbf{x}) = \frac{f_t(\mathbf{x})}{\max_{l=1,\ldots,L} |f_t(\mathbf{x})|}. \tag{13}$$

The final RAB-SLPSVM ensemble output is given by

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \tilde{f}_t(\mathbf{x}), \tag{14}$$

and its sign indicates the decision.

We repeat that, as the experiments will make evident, it is the combination of our new, carefully designed subsampling scheme in order to force diversity and weakness in the SVM learners, and their sparsity inducing LP training the (double) reason for good results in performance and compactness. Our advise is that any other successful approach in the same direction must include the same kind of ingredients.

4. Experiments

We will compare the accuracy and operation computational load of the proposed RAB-LPSVM with those of the SVM and LPSVM classifiers when they are used as single classifiers. Five different boosting ensembles with SVM learners are also considered for comparison purposes. The first two ensembles are the already mentioned D-ABSVM [50] and AB-WSV [52]. RAB-SVM and RAB-LPSVM use SVMs and LPSVMs as learners, training them with the emphasized (but no subsampled) data set. The last approach, RAB-SSVM, uses subsampled versions of standard SVMs as learners. These last three methods will permit to check whether the advantages of our method are due to the use of LPSVM learners, to the subsamplig process, or to their combination.

4.1. Experimental setup

4.1.1. Datasets

Twelve well-known problems will serve for conducting our experiments just to permit an easy appreciation of the relevance of the results, but we

Table 1: Main Characteristics of the Benchmark Problems

Problem	d	L_1/L_{-1} Train	L_1/L_{-1} Test
Abalone (Ab)	8	1238/1269	843/827
Breast (Br)	9	145/275	96/183
Contraceptive (Co)	9	506/377	338/252
Diabetes (Di)	8	172/296	96/204
Duke breast-cancer (Du)	7128	20/23	25/18
Ionosphere (Io)	34	101/100	124/26
Kwok (Kw)	2	300/200	6120/4080
Madelon (Ma)	500	1000/1000	300/300
Ripley (Ri)	2	125/125	500/500
Thyroid (Th)	5	43/97	22/53
Twonorm (Tw)	20	199/201	3504/3496
Waveform (Wa)	21	124/276	1523/3077

select them corresponding to problem of different characteristics (dimension, database size, and difficulty). Ten databases are from the UCI repository [61]: Abalone, Breast, Contraceptive, Diabetes, Duke Breast-cancer, Ionosphere, Madelon, Thyroid, Twonorm, and Waveform. Kwok and Ripley are synthetic problems from [62] and [63], respectively. Table 1 shows their main features (d: dimension; L_1/L_{-1}: number of samples for the training and the test sets). For the sake of brevity, these problems will be denoted with their first two letters.

4.1.2. Machines

We use Gaussian kernels for all the machines. For all the designs but D-ABSVM, their width σ is the same, and its value and that of parameter C are established by means of a 10 run, 5-fold cross-validation (CV) process for subsampling designs and 1 run, 5-fold CV per partition for the rest, exploring 12 values logarithmically spaced in the ranges $[10^{-2}, 10^{5}]$ for C and $\left[10^{-1}\sqrt{d}, 10^{2}\sqrt{d}\right]$ for σ; i.e., $C \in \{0.01, 0.04, 0.19, 0.81, 3.51, 15.2, 65.79, 284.8, 1232.85, 5336.7, 23101.3, 100000\}$ and $\sigma \in \{0.1, 0.19, 0.35, 0.66, 1.23, 2.31, 4.33, 8.1, 15.2, 28.5, 53.4, 100\}\sqrt{d}$. This is a value grid usually explored in standard SVM designs, because it covers wide ranges of values that are enough for most practical cases (note that it can be expanded if necessary: When the best performance is obtained for extreme values). The same CV process serves to select L' for the subsampling designs, considering as possibilities 5%, 10%, 15%, 25%, 50%, and 75% of the total number of training data L. Note that L' is the only additional parameter which our designs require to be cross-validated (with C and σ). The effects of different values of L' are discussed in detail in Subsection 4.3.

The adjustment of the parameters of D-ABSVM is not straightforward: A CV selection tends to provide underperforming ensembles. We have established their values according to [50]: The parameter C is empirically fixed to 50 (we have experimentally verified that exploring other values does not improve results); the initial width σ_{ini} is set as the dispersion radius of the samples in the input space; the minimal width σ_{min} is calculated as the average of the minimum distances between samples; the step size σ_{step} is empirically set to 2; and the diversity threshold has been set to 0.7.

In the case of the AB-WSV ensembles, a weakness constant γ (used to control the learner weakness) must be additionally fixed together with the parameter ν for ν-SVM learners. Following the advice of [52], both parameters are selected by a CV process with 9 equally spaced values in the range $[0.1, 0.9]$, together with parameter σ, which is explored in the same range that for the above-mentioned methods.

The ensemble growth is stopped according to the approach proposed in [38], which selects T as the first value holding

$$\frac{\sum_{t=T-9}^{T} \alpha_t}{\sum_{t=1}^{T} \alpha_t} < T_{stop} \tag{15}$$

where T_{stop} has been empirically set to 0.3 for all the problems and algorithms, except for problem Tw and AB-WSV ensembles, where 0.4 is the value for the threshold parameter.

Finally, it is worth to say that all algorithms have been implemented in MATLAB R2007b and the optimization problems have been solved with MOSEK [64].

4.2. Performance analysis

Table 2 shows the Classification Error (CE) percentage rates and the number of learners building up each ensemble (T) for SVM and LPSVM single classifiers, D-ABSVM and AB-WSV ensembles, and the proposed RAB-SLPSVM algorithm. RAB-SLPSVM results include averaged values (together to standard deviations) calculated over 50 independent runs, because the implicit subsampling mechanism imposes it for a reasonable evaluation, while in the rest of the designs the SVM algorithms lead to a single value (a deterministic solution). So, the comparison consists on checking if

the statistics of a distribution indicate if that distribution is located mainly to the right (bigger) or to the left (smaller) than the single performance value of the other designs. Note that (sample) mean and standard deviation are just the parameters to decide about it. The best results are highlighted in boldface.

Comparing their performances with those of the single classifiers, it is clear that D-ABSVM and AB-WSV ensembles do not solve in a completely satisfactory manner the difficulty to introduce diversity but in a few cases (Ab and Ri for AB-WSV, and, with a slight advantage, Tw and Wa for D-ABSVM). Later, we will see that these ensembles, and D-ABSVM in particular because the different widths of its kernels, have a very high number of SVs, which is a clear practical disadvantage.

On the contrary, the proposed RAB-SLPSVM method offers a better performance than the best single SVM machine in all the problems but Tw and Wa, for which there are ties. RAB-SLPSVM is also better than AB-WSV in all the cases, and better than D-ABSVM but in Tw and Wa again, the differences being very small. The advantage of RAB-SLPSVM is really important for Co, Di, Du, Io, and Ri. Thus, we can conclude that the proposed algorithm successfully solves the diversity difficulty when working with SVM learners.

To analyze the causes of the improved performance of the RAB-SLPSVM ensembles, Table 3 shows the CE and number of learners of its different subversions. SVM and LPSVM results are also included for easier comparisons. All subsampling based methods include averaged values (together to standard deviations) calculated over 50 independent runs. As predictable, Table 3

shows that the direct use of SVM machines for boosting (RAB-SVM, RAB-LPSVM) is not efficient, because these ensembles have better performance than their single counterparts (SVM) in few cases (for Br and Ri, and also for Th when considering the RAB-SVM design). This fact gives evidence of the limitations that the stable character of SVMs creates. Other well known properties can also be observed, such as the sparsity advantage of LP designs and the inferior performance, in general terms, of standard LP formulations.

Now, let us discuss what are the consequences of excluding the use of each one of the elements we combine in order to design high performance ensembles. RAB-LPSVM is the scheme which does not include subsampling. When comparing its results with those of the proposed algorithm, we see ties just for Br, and worse performances for the rest of the problems we are dealing with. Applying the conventional design for SVM base learners (but keeping active the subsampling process), we obtain RAB-SSVM. Note that these ensembles are always worse than RAB-SLPSVM. These evidences support our conjecture of the necessity of an adequate combination of both a sparsity inducing design and an appropriate subsampling to force diversity and weakness in the base learners. Obviously, the overall results also support the carefully selected subsampling method we are proposing.

Finally, a few words about other effects of using LP. This not only provides compact designs, as expected because sparsity is a well-known characteristic of LP optimization procedures -we will discuss the specific result in the next subsection,- but also it seems to produce low dispersion in performance, which is an interesting additional advantageous characteristic.

4.3. Effects of subsampling intensity

Of course, all the above found advantages of the proposed RAB-SLPSVM method are not for free. Additionally to the increase of the training computational effort with respect to the basic SVM or LPSVM, this ensemble design requires CV not only of C and σ, but also of the additional parameter L'. The effects of failing in finding correct values of C and σ are pretty well known: Increasing C reduces sparsity, and increasing σ reduces the generalization capabilities of the machine. Many textbooks elaborate on this, see [5, 15], for example. Here, we will pay attention here to the effect of using different values of L'.

For this analysis, we have included in Table 4 the CE rates (averaged over 50 independent runs), as well as the ensemble sizes, T, for different values of the subsampling parameter L'. Values selected during the CV process are pointed out in boldface.

At the light of these results, the effectiveness of the CV process, which selects the optimum L' value in all the cases, is completely clear. It can also be concluded that a coarse exploration, as that used in our experiments, gives good results with an affordable design computational effort. Furthermore, these results reveals that we could have reduced the explored range to moderately low values of L' (from 10% to 50%), even around to 15%, obtaining similar results with an additional computational saving during the CV process. At the limit, a direct selection of 15% seems to be a good rule-of-thumb, because the results for all the problems but Du are practically optimal.

Regarding the sensitivity of the RAB-SLPSVM algorithm to this parameter, Table 4 shows as slights variations from the cross-validated value cause

minor CE increases in most of the problems. The only cases where the selection of L' seems to be critical are Du and Th, causing a CE increase of around one percent. However, it is important to note that, despite the performance degradation shown in these problems, RAB-SLPSVM would still beat the CE rate of the remaining methods.

4.4. Computational aspects

To analyze the computational load of the proposed SVM ensembles we can exploit the fact that they are monolithic structures and compact common kernels of the different learners into a single one. In this case, as it is shown in Table 5, we can compare the resulting number of kernels of AB-WSV and our RAB-SLPSVM designs with those of single SVM classifiers, because it indicates what is the relative computational effort for classifying unseen samples. We do not include in the corresponding Table 5 the D-ABSVM ensembles in these comparisons because their SVM learners have not kernels of a unique width and, consequently, cannot be compacted, but we clarify that the total number of different centroids in these ensembles is higher than the corresponding numbers for any of the tabulated cases and, consequently, D-ABSVM ensembles require the highest classification computational effort for all the problems we are considering.

The favorable influence of LP based designs appears in RAB-SLPSVM, whose number of different kernels is clearly lower than the numbers for AB-WSV ensembles in all the cases. The increase of N_{SV} in RAB-SLPSVM with respect to LPSVM is at most of one order of magnitude (with an exception for Ab, where it is lower). This is an affordable price to obtain the clearly relevant performance advantages that appear in most the analyzed problems.

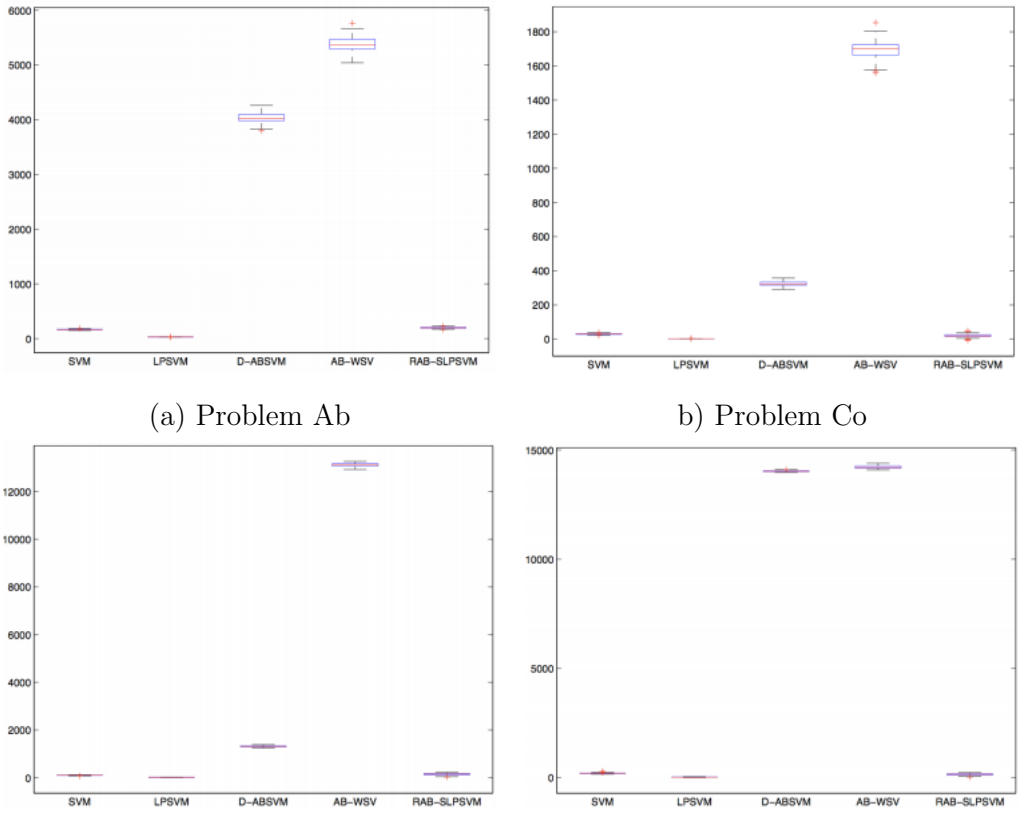
And specially remarkable is the fact that the number of kernels of the compacted RAB-SLPSVM is lower than that of the single standard SVM (more powerful than the single LPSVM) for all the problems but Br, Io, and Wa, for which they are very similar. So, we can say that the combination of the subsampling procedure and the LPSVM training we are proposing not only provides higher performance results via boosting, but, in general terms, more compact arquitectures when compared with the monolithic SVM standard designs.

For the sake of completeness, we have also run the different methods over a 3.0 GHz Intel (R) Xeon (R) E540 computer with 8.0 GB of RAM and we have measured the training (t_r) and operation (t_o) times (milliseconds). For comparison reasons, all the methods have been implemented with Matlab using the MOSEK (Mosek, 2010) toolbox as the optimization tool and the experiment has been repeated over 100 different runs. Figures 1 and 2 show the boxplot of these times (so that averaged values and their standard desviations are included) for the different methods under study in four representative problems (Ab, Co, Kw, Ma).

On the one hand, classification time conclusions are similar to those revealed by Table 5: RAB-SLPSVM architecture requires computational times similar to those of standard LP-SVM and SVM classifiers. Furthermore, we can now check the increased burden of D-ABSVM ensembles due to its incapacity of fusing kernels.

On the other hand, training times are as expected, i.e., ensemble method training time increases with the number of learners. However, due to the fact they are using simplest learners, in some cases (see Kw or Ma) these

times are similar or, even, lower than those of LPSVM classifiers. Finally, it is important to note that these times are in milliseconds, so the training of any of these approaches is completely affordable, and the differences are not significant.



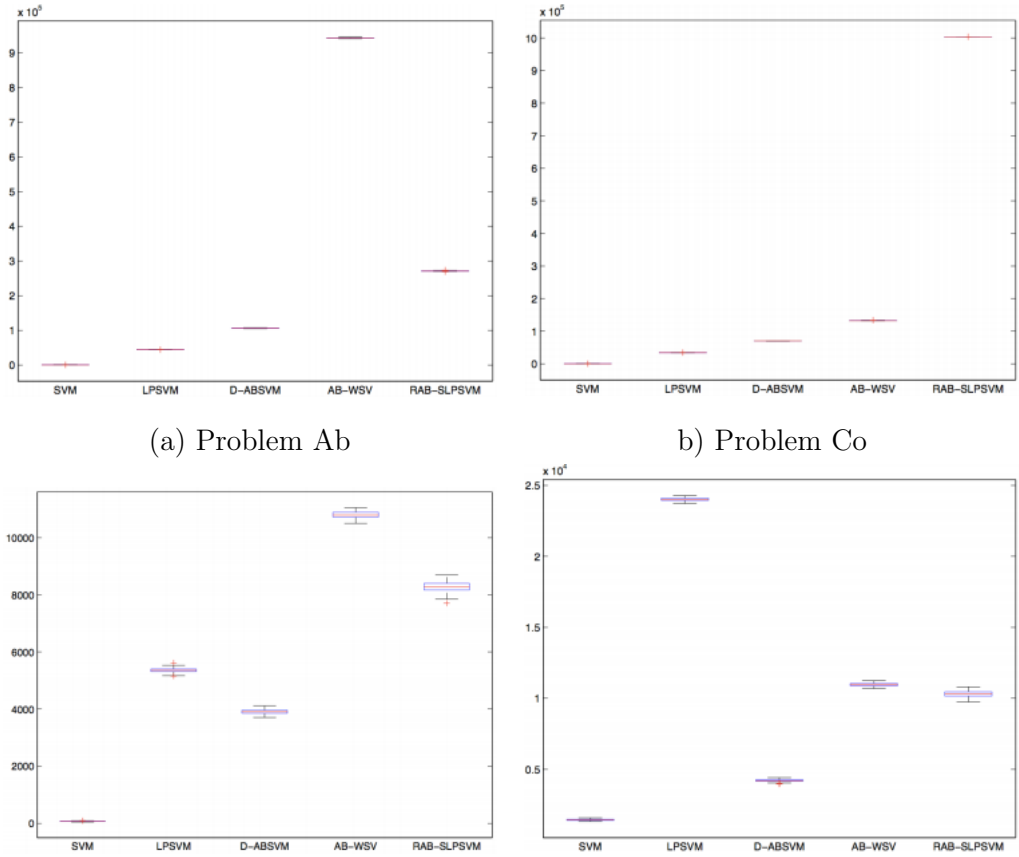
(a) Problem Ab

b) Problem Co

(c) Problem Kw

d) Problem Ma

Figure 1: Boxplot of the averaged operation times (ms), including their standard deviations, in four representative problems for all the ensembles under study.



(a) Problem Ab

b) Problem Co

(c) Problem Kw

d) Problem Ma

Figure 2: Boxplot of the averaged training times (ms), including their standard deviations, in four representative problems for all the ensembles under study.

5. Conclusions

In order to built high performance and compact SVM designs by means of Real AdaBoost ensembles, we propose in this paper a combination of a new and carefully designed iteratively re-stratified subsampling procedure – to force diverse and weak learners– and a dual domain LP training of learners

–to keep sparsity under control–. Experiments conducted with a number of well-known benchmark databases provide evidence of the advantages of the proposed method, both in performance when compared with single SVM classifiers and with selected previous designs of RAB SVM ensembles, and in compactness with respect to these ensembles and even to the standard single SVM designs.

The price for getting these advantages is a higher computational demand for designing the ensemble, which can be qualified of affordable and that can be further reduced given the relative insensitivity of the proposed designs to the subsampling intensity.

To explore how to extend the proposed procedure to other kinds of maximal margin trainable learners and to other boosting algorithms is a promising avenue for further research.

Acknowledgements

This work was supported in part by the Spanish MICINN under Grants TEC 2011-22480 and TIN 2011-24533.

References

[1] V. Vapnik, Estimation of Dependences Based on Empirical Data, Springer-Verlag, Secaucus, NJ, USA, 1982.

[2] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, NY, 1995.

[3] V. Vapnik, Statistical Learning Theory, Wiley, New York, NY, 1998.

[4] G. Kimeldorf, G. Wahba, Some results on Tchebycheffian spline functions, J. Mathematical Analysis and Applications 33 (1971) 82–95.

[5] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, 2002.

[6] R. Herbrich, Learning Kernel Classifiers: Theory and Algorithms, MIT Press, Cambridge, MA, USA, 2001.

[7] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, NY, 2004.

[8] L. Wang, Support Vector Machines: Theory and Applications, Springer, New York, NY, 2005.

[9] I. Steinwart, A. Christmann, Support Vector Machines, Springer, New York, NY, 2008.

[10] J. Shawe-Taylor, S. Sun, A review of optimization methodologies in support vector machines, Neurocomputing 74 (17) (2011) 3609–3618.

[11] J. L. Rojo-Álvarez, M. Martínez-Ramón, A. R. Figueiras-Vidal, A. García-Armada, A. Artés-Rodríguez, A robust support vector algorithm for nonparametric spectral analysis, IEEE Signal Processing Letters 10 (2003) 320–323.

[12] J. L. Rojo-Álvarez, M. Martínez-Ramón, M. de Prado-Cumplido, A. Artés-Rodríguez, A. R. Figueiras-Vidal, Support vector method for

robust ARMA system identification, IEEE Trans. on Signal Processing 52 (2004) 155–164.

[13] J. L. Rojo-Álvarez, G. Camps-Valls, M. Martínez-Ramón, E. Soria-Olivas, Á. Navia-Vázquez, A. R. Figueiras-Vidal, Support vector machines framework for linear signal processing, Signal Processing 85 (12) (2005) 2316–2326.

[14] M. Martínez-Ramón, J. L. Rojo-Álvarez, G. Camps-Valls, J. Muñoz-Marí, Á. Navia-Vázquez, E. Soria-Olivas, A. R. Figueiras-Vidal, Support vector machines for nonlinear kernel ARMA system identification, IEEE Trans. on Neural Networks 17 (2006) 1617–1622.

[15] C. M. Bishop, Pattern Recognition and Machine Learning (p. 332), Springer, New York, NY, 2006.

[16] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9 (1999) 293–300.

[17] P. S. Bradley, O. L. Mangasarian, Feature selection via concave minimization and support vector machines, in: Proc. 15th Intl. Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 82–90.

[18] V. Kecman, I. Hadzic, Support vectors selection by linear programming, in: Proc. Intl. Joint Conf. on Neural Networks, Vol. 5, Como, Italy, 2000, pp. 193–198.

[19] W. Zhou, L. Zhang, L. Jiao, Linear programming support vector machines, Pattern Recognition 35 (2002) 2927 – 2936.

[20] J. Zhu, S. Rosset, T. Hastie, R. Tibshirani, 1-norm support vector machines, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Proc. Sys. 16, MIT Press, Cambridge, MA, 2004, pp. 49–56.

[21] S. Zhou, Which is better? Regularization in RKHS vs R^m on reduced SVMs, Statistics, Optimization and Information Computing 1 (2013) 82–106.

[22] S. El Jelali, A. Lyhyaoui, A. R. Figueiras-Vidal, Designing model based classifiers by emphasizing soft targets, Fundam. Inf. 96 (2009) 419–433.

[23] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks 12 (1999) 1399–1404.

[24] Y. Liu, X. Yao, Simultaneous training of negatively correlated neural networks in an ensemble, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 29 (1999) 716–725.

[25] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Computer Sys. Sciences 55 (1997) 119 – 139.

[26] R. E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (1999) 297–336.

[27] R. E. Schapire, Y. Freund, Boosting: Foundations and Algorithms, MIT Press, Cambridge, MA, 2012.

[28] H. Drucker, R. E. Schapire, P. Simard, Boosting performance in neural

networks, Intl. J. of Pattern Recognition and Artificial Intelligence 7 (1993) 705–719.

[29] Y. LeCun, L. D. Jackel, H. A. Eduard, N. Bottou, C. Cortes, J. S. Denker, H. Drucker, E. Sackinger, P. Simard, V. Vapnik, Learning algorithms for classification: A comparison on handwritten digit recognition, in: J. H. Oh, C. Kwon, S. Cho (Eds.), Neural Networks: The Statistical Mechanics Perspective, World Scientific, Singapore, 1995, pp. 261–276.

[30] H. Schwenk, Y. Bengio, Adaboosting neural networks, in: W. Gerstner, A. Germond, M. Hasler, J. D. Nicoud (Eds.), Proc. 7th Intl. Conf. on Artificial Neural Networks (LNCS 1327), Springer, Berlin, 1997, pp. 967–972.

[31] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning 36 (1999) 105–139.

[32] T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, Machine Learning 40 (2000) 139–157.

[33] L. Breiman, Prediction games and arcing algorithms, Neural Computation 11 (1999) 1493–1517.

[34] G. Rätsch, T. Onoda, K. R. Müller, Soft margins for AdaBoost, Machine Learning 42 (2001) 287–320.

[35] G. Rätsch, M. K. Warmuth, Efficient margin maximizing with boosting, J. Machine Learning Res. 6 (2005) 2131–2152.

[36] Y. Sun, S. Todorovic, J. Li, Reducing the overfitting of AdaBoost by controlling its data distribution skewness, Intl. J. Pattern Recognition and Artificial Intelligence 20 (2006) 1093–1116.

[37] V. Gómez-Verdejo, M. Ortega-Moral, J. Arenas-García, A. R. Figueiras-Vidal, Boosting by weighting critical and erroneous samples, Neurocomputing 69 (2006) 679 – 685.

[38] V. Gómez-Verdejo, J. Arenas-García, A. R. Figueiras-Vidal, A dynamically adjusted mixed emphasis method for building boosting ensembles, IEEE Trans. Neural Networks 19 (2008) 3 –17.

[39] C.-X. Zhang, J.-S. Zhang, G.-Y. Zhang, An efficient modified boosting method for solving classification problems, J. Computational and Applied Mathematics 214 (2008) 381 – 392.

[40] C. Shen, H. Li, Boosting through optimization of margin distributions, IEEE Trans. Neural Networks 21 (4) (2010) 659–666.

[41] A. Aachad, A. Omari, A. R. Figueiras-Vidal, Neighborhood guided smoothed emphasis for real adaboost ensembles, submitted for publication to Neural Proc. Letters.

[42] J. Wickramaratna, S. Holden, B. Buxton, Performance degradation in boosting, in: J. Kittler, F. Roli (Eds.), Multiple Classifier Systems (LNCS 2096), Springer, Berlin, 2001, pp. 11–21.

[43] M. Kawakita, S. Eguchi, Boosting method for local learning in statistical pattern recognition, Neural Computation 20 (2008) 2792–2838.

[44] E. Mayhua-López, V. Gómez-Verdejo, A. R. Figueiras-Vidal, Real AdaBoost with gate controlled fusion, IEEE Trans. Neural Networks and Learning Sytems 23 (12) (2012) 2003 –2009.

[45] A. Omari, A. R. Figueiras-Vidal, Feature combiners with gate generated weights for classification, IEEE Trans. Neural Networks and Learning Systems 24 (2013) 158–163.

[46] A. Omari, A. R. Figueiras-Vidal, Ensemble post-aggregation by means of a gate-generated functional weight classifiers, Submitted to Information Fusion.

[47] W. Wu, Z. Yanan, W. Linlin, An AdaBoost algorithm with SVM based on nonlinear decision function, in: Proc. Intl. Conf. on Computational Intell. and Natural Computing, Wuhan, China, 2009, pp. 22–25.

[48] N. Lima, A. Neto, J. de Melo, Creating an ensemble of diverse support vector machines using AdaBoost, in: Proc. Intl. Joint Conf. on Neural Networks, Atlanta, GA, USA, 2009, pp. 1802 –1806.

[49] T. Wei, Z. Qin, X. Cao, B. Leng, A boosting method based on SVM for relevance feedback in content-based 3D model retrieval, in: Proc. 2nd Intl. Conf. on Software Engineering and Data Mining, Chengdu, China, 2010, pp. 517–522.

[50] X. Li, L. Wang, E. Sung, AdaBoost with SVM-based component classifiers, Eng. Appl. Artif. Intell. 21 (2008) 785–795.

[51] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, S. Y. Bang, Constructing support vector machine ensemble, Pattern Recognition 36 (2003) 2757 – 2767.

[52] P. Rangel, F. Lozano, E. García, Boosting of support vector machines with application to editing, in: Proc. 4th Intl. Conf. on Machine Learning and Applications, Los Angeles, CA, USA, 2005, p. 6.

[53] S. Muñoz Romero, J. Arenas-García, V. Gómez-Verdejo, Real AdaBoost ensembles with emphasized subsampling, in: J. Cabestany, F. Sandoval, A. Prieto, J. Corchado (Eds.), Bio-Inspired Systems: Computational and Ambient Intelligence (LNCS 5517), Springer, Berlin, 2009, pp. 440–447.

[54] M. Villamizar, J. Andrade-Cetto, A. Sanfeliu, F. Moreno-Noguer, Bootstrapping boosted random ferns for discriminative and efficient object classification, Pattern Recognition 45 (2012) 3141 – 3153.

[55] Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines. Theory and application to the classification of microarray data and satellite radiance data, J. of the American Statistical Assoc. 99 (2004) 67–81.

[56] Y. Ji, S. Sun, Multitask multiclass support vector machines: Model and experiments, Pattern Recognition 46 (3) (2013) 914–924.

[57] J. Zhu, H. Zou, S. Rosset, T. Hastie, Multi-class AdaBoost, Statistics and Its Interface 2 (2009) 349–360.

[58] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, Hoboken, NJ, 2004.

[59] L. Rokach, Pattern Classification Using Ensemble Methods, World Scientific, Singapore, 2010.

[60] Z. Kalal, J. Matas, K. Mikolajczyk, Weighted sampling for large-scale boosting, in: M. Everingham, C. J. Needham, R. Fraile (Eds.), Proc. of the British Machine Vision Conference, British Machine Vision Assoc., Leeds, UK, 2008, pp. 42.1–42.10.

[61] A. Frank, A. Asuncion, UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences (2010). URL `http://archive.ics.uci.edu/ml`

[62] J. Y. Kwok, Moderating the outputs of support vector machine classifiers, IEEE Trans. Neural Networks 10 (1999) 1018–1031.

[63] B. D. Ripley, Neural networks and related methods for classification, J. Royal Statistical Society 56 (1994) 409–456.

[64] Mosek, The MOSEK Optimization Tools Manual Version 6.0 (Revision 66), MOSEK ApS, Copenhagen, Denmark, 2010.

Table 2: Classification Error (CE) percentage rates and number of learners (T) provided by the algorithms SVM, LPSVM, D-ABSVM, AB-WSV, and RAB-SLPSVM.

		SVM	**LPSVM**	**D-ABSVM**	**AB-WSV**	**RAB-SLPSVM**
Ab	$CE(\%)$	20.1	20.0	22.4	19.4	**19.1** $\pm$ 0.1
	T	–	–	51.0	3.0	13.9 $\pm$ 1.8
Br	$CE(\%)$	2.5	2.5	2.5	4.3	**2.2** $\pm$ 0.2
	T	–	–	69.0	7.0	10.1 $\pm$ 0.8
Co	$CE(\%)$	28.3	28.6	33.9	35.3	**27.9** $\pm$ 0.1
	T	–	–	65.0	9.0	15.1 $\pm$ 0.5
Di	$CE(\%)$	19.3	22.3	20.7	20.3	**18.2** $\pm$ 0.3
	T	–	–	95.0	7.0	21.0 $\pm$ 0.0
Du	$CE(\%)$	13.9	14.0	15.3	14.0	**12.0** $\pm$ 1.2
	T	–	–	73.0	4.0	17.3 $\pm$ 0.6
Io	$CE(\%)$	**2.0**	**2.0**	4.7	9.3	**2.0** $\pm$ 0.7
	T	–	–	191	8.0	23.5 $\pm$ 4.0
Kw	$CE(\%)$	11.8	11.7	13.1	12.2	**11.5** $\pm$ 0.0
	T	–	–	10.0	2.0	16.9 $\pm$ 0.6
Ma	$CE(\%)$	42.2	43.5	44.2	41.8	**40.8** $\pm$ 0.6
	T	–	–	73.0	14.0	9.0 $\pm$ 0.8
Ri	$CE(\%)$	9.5	9.4	10.8	9.2	**8.8** $\pm$ 0.2
	T	–	–	22.0	8.0	13.9 $\pm$ 0.7
Th	$CE(\%)$	5.3	6.7	12.0	5.3	**4.5** $\pm$ 0.8
	T	–	–	91.0	2.0	14.8 $\pm$ 2.3
Tw	$CE(\%)$	2.5	2.8	**2.4**	3.2	2.5 $\pm$ 0.1
	T	–	–	2.0	3.0	6.6 $\pm$ 0.5
Wa	$CE(\%)$	10.5	11.2	**10.4**	13.1	10.5 $\pm$ 0.1
	T	–	–	99.0	6.0	25.0 $\pm$ 0.1

Table 3: Performance analysis of the proposed RAB-SLPSVM method together to its intermediate versions: RAB-SVM, RAB-SSVM and RAB-LPSVM, and the single SVM and LPSVM classifiers. $CE(\%)$: Classification error percentage rates (average $\pm$ standard deviation); T: Number of learners (average $\pm$ standard deviation)

		SVM	**LPSVM**	**RAB-SVM**	**RAB-LPSVM**	**RAB-SSVM**	**RAB-SLPSVM**
Ab	$CE(\%)$	20.1	20.0	21.4	21.7	20.9 ± 0.4	**19.1** ± 0.1
	T	–	–	12.0	18.0	8.5 ± 1.0	13.9 ± 1.8
Br	$CE(\%)$	2.5	2.5	**2.2**	**2.2**	3.2 ± 0.5	**2.2** ± 0.2
	T	–	–	8.0	10.0	9.6 ± 2.4	10.1 ± 0.8
Co	$CE(\%)$	28.3	28.6	30.2	28.8	30.5 ± 1.4	**27.8** ± 0.1
	T	–	–	15.0	14.0	12.0 ± 1.3	10.2 ± 0.5
Di	$CE(\%)$	19.3	22.3	20.7	26.0	22.3 ± 1.4	**18.2** ± 0.3
	T	–	–	9.0	16.0	7.9 ± 1.4	21.0 ± 0.0
Du	$CE(\%)$	13.9	14.0	14.5	14.2	14.1 ± 1.1	**12.0** ± 1.2
	T	–	–	16	19	17.8 ± 1.2	17.3 ± 0.6
Io	$CE(\%)$	**2.0**	**2.0**	**2.0**	2.7	3.5 ± 1.2	**2.0** ± 0.7
	T	–	–	18.0	18.0	11.4 ± 1.0	23.5 ± 4.0
Kw	$CE(\%)$	11.8	11.7	11.8	11.8	13.2 ± 1.1	**11.5** ± 0.0
	T	–	–	13.0	15.0	7.6 ± 1.4	16.9 ± 0.6
Ma	$CE(\%)$	42.2	43.5	43.6	44.7	42.3 ± 0.9	**40.8** ± 0.6
	T	–	–	20	21	17.1 ± 1.2	9.0 ± 0.8
Ri	$CE(\%)$	9.5	9.4	**8.8**	9.2	11.8 ± 2.4	**8.8** ± 0.2
	T	–	–	7.0	12.0	9.6 ± 2.1	13.9 ± 0.7
Th	$CE(\%)$	5.3	6.7	5.0	6.7	4.7 ± 1.1	**4.5** ± 0.8
	T	–	–	20.0	13.0	22.2 ± 1.6	14.8 ± 2.3
Tw	$CE(\%)$	2.5	2.8	2.5	3.5	4.3 ± 1.4	2.5 ± 0.1
	T	–	–	7.0	11.0	18.9 ± 2.9	6.6 ± 0.5
Wa	$CE(\%)$	10.5	11.2	**10.4**	11.3	11.2 ± 0.2	10.5 ± 0.1
	T	–	–	14.0	16.0	14.5 ± 0.6	25.0 ± 0.1

Table 4: Classification Error (CE) percentage rates and number of learners (T) provided by the RAB-SLPSVM algorithm when different values of L' are used.

		5%	10%	15%	25%	50%	75%	100%
Ab	CE	21.9 ± 0.3	20.9 ± 0.2	**19.1** ± 0.1	19.5 ± 0.1	21.2 ± 0.1	21.5 ± 0.1	22.1 ± 0.1
	T	20.3 ± 0.9	18.3 ± 0.6	13.9 ± 0.1	17.1 ± 0.8	20.1 ± 0.3	21.3 ± 0.4	20.2 ± 0.3
Br	CE	2.4 ± 0.3	**2.2** ± 0.2	2.2 ± 0.2	2.3 ± 0.2	2.4 ± 0.2	2.4 ± 0.1	2.5 ± 0.1
	T	20.0 ± 0.1	10.1 ± 0.8	14.3 ± 0.7	14.2 ± 0.9	16.3 ± 0.9	14.5 ± 0.7	15.2 ± 0.9
Co	CE	29.1 ± 0.4	28.7 ± 0.4	**27.9** ± 0.1	28.8 ± 0.3	29.1 ± 0.1	29.2 ± 0.1	29.2 ± 0.1
	T	15.1 ± 1.2	10.3 ± 1.4	15.1 ± 0.5	13.5 ± 1.0	10.2 ± 1.5	11.2 ± 1.3	10.1 ± 1.5
Di	CE	19.3 ± 0.5	18.5 ± 0.4	**18.2** ± 0.3	19.1 ± 0.6	19.5 ± 0.2	19.7 ± 0.2	20.3 ± 0.2
	T	20.0 ± 0.7	25.1 ± 1.3	21.0 ± 0.0	20.1 ± 0.1	22.3 ± 0.2	21.3 ± 0.1	37.1 ± 1.7
Du	CE	17.3 ± 1.2	17.2 ± 1.2	15.3 ± 1.3	14.2 ± 1.0	**12.0** ± 1.2	13.3 ± 1.2	16.9 ± 1.4
	T	25.2 ± 0.7	30.5 ± 0.3	29.1 ± 0.9	15.0 ± 1.2	17.3 ± 0.6	18.5 ± 1.7	19.3 ± 1.2
Io	CE	2.3 ± 0.9	2.2 ± 0.9	**2.0** ± 0.7	2.1 ± 0.7	2.3 ± 0.8	2.5 ± 0.9	2.7 ± 0.9
	T	26.3 ± 4.5	22.1 ± 3.7	23.5 ± 4.0	22.6 ± 3.0	23.0 ± 2.5	22.5 ± 2.3	21.1 ± 3.2
Kw	CE	11.9 ± 0.1	11.7 ± 0.0	**11.5** ± 0.0	11.7 ± 0.0	11.8 ± 0.0	11.8 ± 0.0	11.9 ± 0.0
	T	17.1 ± 0.9	18.2 ± 0.5	16.9 ± 0.6	20.0 ± 0.5	19.3 ± 0.9	21.4 ± 0.5	20.7 ± 0.3
Ma	CE	45.3 ± 0.7	42.4 ± 0.8	**40.8** ± 0.6	41.5 ± 0.7	43.2 ± 0.3	44.5 ± 0.2	44.9 ± 0.1
	T	12.3 ± 0.9	14.5 ± 0.7	9.0 ± 0.8	17.3 ± 0.2	10.3 ± 0.3	12.4 ± 0.1	12.1 ± 0.3
Ri	CE	9.5 ± 0.4	9.2 ± 0.2	8.9 ± 0.2	**8.8** ± 0.2	9.3 ± 0.1	9.5 ± 0.1	9.7 ± 0.1
	T	16.2 ± 0.9	15.3 ± 0.5	14.3 ± 0.85	13.9 ± 0.7	18.0 ± 0.9	19.3 ± 0.5	19.0 ± 0.7
Th	CE	6.7 ± 1.2	5.1 ± 1.2	**4.5** ± 0.8	6.2 ± 1.0	5.7 ± 0.8	5.9 ± 0.8	6.3 ± 0.9
	T	16.3 ± 2.9	15.4 ± 2.0	14.8 ± 2.3	17.3 ± 2.5	18.4 ± 2.2	17.3 ± 2.1	20.1 ± 4.2
Tw	CE	2.7 ± 0.4	**2.5** ± 0.1	2.6 ± 0.1	2.6 ± 0.1	2.7 ± 0.1	3.0 ± 0.1	3.2 ± 0.1
	T	7.9 ± 0.3	6.6 ± 0.5	8.3 ± 0.9	7.5 ± 0.3	8.2 ± 0.2	9.0 ± 0.7	9.3 ± 0.2
Wa	CE	12.1 ± 0.4	10.9 ± 0.1	**10.5** ± 0.1	10.7 ± 0.1	10.9 ± 0.1	11.0 ± 0.1	11.3 ± 0.1
	T	25.0 ± 0.2	26.3 ± 0.4	25.0 ± 0.1	23.4 ± 0.2	24.5 ± 0.5	26.2 ± 0.5	25.2 ± 0.9

Table 5: Number of different kernels (N_{SV}) making up the SVM and LPSVM classifiers and the AB-WSV and RAB-SLPSVM ensembles.

	SVM	LPSVM	AB-WSV	RAB-SLPSVM
Ab	1175	59	1374	16 ± 3
Br	50	4	278	59 ± 5
Co	578	27	710	265 ± 48
Di	264	31	275	94 ± 6
Du	43	19	36	36 ± 3
Io	121	26	104	170 ± 7
Kw	132	14	447	129 ± 7
Ma	186	77	1249	179 ± 10
Ri	91	16	183	88 ± 6
Th	85	20	100	65 ± 9
Tw	117	12	61	51 ± 4
Wa	121	16	271	136 ± 7