Benchmarking Least Squares Support Vector Machine Classifiers

TONY VAN GESTEL, JOHAN A.K. SUYKENS†,
Dept. of Electrical Engineering - ESAT/SISTA, Katholieke Universiteit Leuven, Belgium, E-mail: {tony.vangestel,johan.suykens}@esat.kuleuven.ac.be

BART BAESENS, STIJN VIAENE,
Leuven Institute for Research on Information Systems, Katholieke Universiteit Leuven, Belgium E-mail: {bart.baesens,stijn.viaene}@econ.kuleuven.ac.be

JAN VANTHIENEN, GUIDO DEDENE,
Leuven Institute for Research on Information Systems, Katholieke Universiteit Leuven, Belgium E-mail: {jan.vanthienen,guido.dedene}@econ.kuleuven.ac.be

BART DE MOOR AND JOOS VANDEWALLE
Dept. of Electrical Engineering - ESAT/SISTA, Katholieke Universiteit Leuven, Belgium, E-mail: {bart.demoor,joos.vandewalle}@esat.kuleuven.ac.be

Abstract. In Support Vector Machines (SVMs), the solution of the classification problem is characterized by a (convex) quadratic programming (QP) problem. In a modified version of SVMs, called Least Squares SVM classifiers (LS-SVMs), a least squares cost function is proposed so as to obtain a linear set of equations in the dual space. While the SVM classifier has a large margin interpretation, the LS-SVM formulation is related in this paper to a ridge regression approach for classification with binary targets and to Fisher's linear discriminant analysis in the feature space. Multiclass categorization problems are represented by a set of binary classifiers using different output coding schemes. While regularization is used to control the effective number of parameters of the LS-SVM classifier, the sparseness property of SVMs is lost due to the choice of the 2-norm. Sparseness can be imposed in a second stage by gradually pruning the support value spectrum and optimizing the hyperparameters during the sparse approximation procedure. In this paper, twenty public domain benchmark datasets are used to evaluate the test set performance of LS-SVM classifiers with linear, polynomial and radial basis function (RBF) kernels. Both the SVM and LS-SVM classifier with RBF kernel in combination with standard cross-validation procedures for hyperparameter selection achieve comparable test set performances. These SVM and LS-SVM performances are consistently very good when compared to a variety of methods described in the literature including decision tree based algorithms, statistical algorithms and instance based learning methods. We show on ten UCI datasets that the LS-SVM sparse approximation procedure can be successfully applied.

Keywords: Least Squares Support Vector Machines, Multiclass Support Vector Machines, Sparse Approximation

† Corresponding author

1. Introduction

Recently, support vector machines (SVMs) have been introduced by Vapnik [7, 54] for solving classification and nonlinear function estimation problems [11, 37, 38, 40, 41, 42, 54, 55, 56]. Within this new approach the training problem is reformulated and represented in such a way so as to obtain a (convex) quadratic programming (QP) problem. The solution to this QP problem is global and unique. In SVMs, it is possible to choose several types of kernel functions including linear, polynomial, RBFs, MLPs with one hidden layer and splines, as long as the Mercer condition is satisfied. Furthermore, bounds on the generalization error are available from statistical learning theory [11, 41, 54, 55], which are expressed in terms of the VC (Vapnik-Chervonenkis) dimension. An upper bound on this VC dimension can be computed by solving another QP problem. Despite the nice properties of SVMs, there are still a number of drawbacks concerning the selection of hyperparameters and the fact that the size of the matrix involved in the QP problem is directly proportional to the number of training points.

A modified version of SVM classifiers, Least Squares SVMs (LS-SVMs) classifiers, was proposed in [44]. A two-norm was taken with equality instead of inequality constraints so as to obtain a linear set of equations instead of a QP problem in the dual space. In this paper, it is shown that these modifications of the problem formulation implicitly correspond to a ridge regression formulation with binary targets ± 1. In this sense, the LS-SVM formulation is related to regularization networks [17, 40], Gaussian Processes regression [58] and to the ridge regression type of SVMs for nonlinear function estimation [36, 41], but with the inclusion of a bias term which has implications towards algorithms. The primal-dual formulations of the SVM framework and the equality constraints of the LS-SVM formulation allow to make extensions towards recurrent neural networks [48] and nonlinear optimal control [49]. In this paper, the additional insight of the linear decision line in the feature space is used to relate the implicit LS-SVM regression approach to a regularized form of Fisher's discriminant analysis [5, 16, 18] in the feature space [3, 30]. By applying the Mercer condition, this so-called Kernel Fisher Discriminant is obtained as the solution to a generalized eigenvalue problem [3, 30].

The QP-problem of the corresponding SVM formulation is typically solved by Interior Point (IP) methods [11, 41], Sequential Minimal Optimization (SMO) [32] and iteratively reweighted least squares approaches (IRWLS) [31], while LS-SVMs [44, 47, 52, 53, 57] result into a set of linear equations. Efficient iterative methods for solving large scale linear systems are available in numerical linear algebra [21]. In

[45] a conjugate gradient based iterative method has been developed for solving the related Karush-Kuhn-Tucker system. In this paper, the algorithm is further refined in order to combine it with hyperparameter selection. A drawback of LS-SVMs is that sparseness is lost due the choice of a 2-norm. However, this can be circumvented in a second stage by a pruning procedure which is based upon removing training points guided by the sorted support value spectrum and does not involve the computation of inverse Hessian matrices as in classical neural network pruning methods [47]. This is important in view of an equivalence between sparse approximation and SVMs as shown in [20].

A straightforward extension of LS-SVMs to multiclass problems has been proposed in [46], where additional outputs are taken in order to encode multi classes as is often done in classical neural networks methodology [5]. This approach is different from standard multiclass SVM approaches. In [46] the multi two-spiral benchmark problem, which is known to be hard for classical neural networks, was solved with an LS-SVM with a minimal number of bits in the class coding and yielded an excellent generalization performance. This approach is further extended here to different types of output codings for the classes, like one-versus-all, error correcting and one-versus-one codes.

In this paper, we report the performance of LS-SVMs on twenty public domain benchmark datasets [6]. Ten binary and ten multiclass classification problems were considered. On each dataset linear, polynomial and RBF kernels were trained and tested. The performances of the simple minimum output coding and advanced one-versus-one coding are also compared on the multiclass problems. The LS-SVM hyperparameters were determined from a 10-fold cross-validation procedure by grid search in the hyperparameter space. The LS-SVM generalization ability was estimated on independent test sets in all of the cases and results of additional randomizations on all datasets are reported. In the experimental setup two third of the data was used to construct the classifier and one third was reserved for out-of-sample testing. After optimization of the hyperparameters, a very good test set performance is achieved by LS-SVMs on all twenty datasets. At this point comparisons on the same randomized datasets have been made with reference classifiers including the SVM classifier, decision tree based algorithms, statistical algorithms and instance based learning methods [1, 9, 16, 24, 25, 28, 35, 59]. In most cases SVMs and LS-SVMs with RBF kernel perform at least as good as SVM and LS-SVMs with linear kernel which means that often additional insight can be obtained whether the optimal decision boundary is strongly nonlinear or not. Furthermore, we successfully confirm the LS-SVM sparse approximation procedure that has been proposed in [47] on 10 UCI benchmark datasets with optimal hyperparameter determination during pruning.

This paper is organized as follows. In Section 2, we review the basic LS-SVM formulation for binary classification and relate the LS-SVM formulation to Kernel Fisher Discriminant Analysis. The multiclass categorization problem is reformulated as a set of binary classification problems in Section 3. The sparse approximation procedure is discussed in Section 4. A conjugate gradient algorithm for large scale applications is discussed in Section 5. Section 6 elaborates on the selection of the hyperparameters with 10-fold cross-validation. Section 7 presents the empirical findings of LS-SVM classifiers achieved on 20 UCI benchmark datasets in comparison with other methods.

2. LS-SVMs for binary classification

Given a training set $\{(x_i, y_i)\}_{i=1}^N$ with input data $x_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier, according to Vapnik's original formulation [7, 11, 37, 38, 40, 54, 55, 56], satisfies the following conditions:

$$\begin{cases} w^T \varphi(x_i) + b \geq +1 \ , & \text{if} \ \ y_i = +1 \\ w^T \varphi(x_i) + b \leq -1 \ , & \text{if} \ \ y_i = -1 \end{cases} \tag{1}$$

which is equivalent to

$$y_i[w^T \varphi(x_i) + b] \geq 1, \quad i = 1, ..., N. \tag{2}$$

The nonlinear function $\varphi(\cdot)$: $\mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$ maps the input space to a high (and possibly infinite) dimensional feature space. In primal weight space the classifier then takes the form

$$y(x) = \text{sign}[w^T \varphi(x) + b], \tag{3}$$

but, on the other hand, is never evaluated in this form. One defines the optimization problem:

$$\min_{w,b,\xi} \mathcal{J}(w, \xi) = \frac{1}{2} w^T w + c \sum_{i=1}^{N} \xi_i \tag{4}$$

subject to

$$\begin{cases} y_i[w^T \varphi(x_i) + b] \geq 1 - \xi_i, & i = 1, ..., N \\ \xi_i \geq 0, & i = 1, ..., N. \end{cases} \tag{5}$$

The variables ξ_i are slack variables which are needed in order to allow misclassifications in the set of inequalities (e.g., due to overlapping distributions). The positive real constant c should be considered as a tuning parameter in the algorithm. For nonlinear SVMs, the QP-problem and the classifier are never solved and evaluated in this form. Instead a dual space formulation and representation is obtained by applying the Mercer condition, see [11, 37, 38, 41, 54, 55, 56] for details.

Vapnik's SVM classifier formulation was modified in [44] into the following LS-SVM formulation:

$$\min_{w,b,e} \mathcal{J}(w,e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{i=1}^{N} e_i^2 \tag{6}$$

subject to the equality constraints

$$y_i \, [w^T \varphi(x_i) + b] = 1 - e_i, \; i = 1, ..., N. \tag{7}$$

This formulation consists of equality instead of inequality constraints and takes into account a squared error with regularization term similar to ridge regression.

The solution is obtained after constructing the Lagrangian:

$$\mathcal{L}(w, b, e; \alpha) = \mathcal{J}(w, b, e) - \sum_{i=1}^{N} \alpha_i \{ y_i [w^T \varphi(x_i) + b] - 1 + e_i \}, \tag{8}$$

where $\alpha_i \in \mathbb{R}$ are the Lagrange multipliers that can be positive or negative in the LS-SVM formulation. From the conditions for optimality, one obtains the Karush-Kuhn-Tucker (KKT) system:

$$\left\{ \begin{array}{ll} \frac{\partial \mathcal{L}}{\partial w} = 0 \; \rightarrow \; w = \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i) & \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \; \rightarrow \; \sum_{i=1}^{N} \alpha_i y_i = 0 & \\ \frac{\partial \mathcal{L}}{\partial e_i} = 0 \; \rightarrow \; \alpha_i = \gamma e_i, & i = 1, ..., N \\ \frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \; \rightarrow \; y_i [w^T \varphi(x_i) + b] - 1 + e_i = 0, & i = 1, ..., N. \end{array} \right. \tag{9}$$

Note that sparseness is lost which is clear from the condition $\alpha_i = \gamma e_i$. As in standard SVMs, we never calculate w nor $\varphi(x_i)$. Therefore, we eliminate w and e yielding [44]

$$\left[\begin{array}{c|c} 0 & y^T \\ \hline y & \Omega + \gamma^{-1} I \end{array} \right] \left[\begin{array}{c} b \\ \hline \alpha \end{array} \right] = \left[\begin{array}{c} 0 \\ \hline 1_v \end{array} \right] \tag{10}$$

with $y = [y_1; ...; y_N]$, $1_v = [1; ...; 1]$, $e = [e_1; ...; e_N]$, $\alpha = [\alpha_1; ...; \alpha_N]$. Mercer's condition is applied within the Ω matrix

$$\Omega_{ij} = y_i y_j \, \varphi(x_i)^T \varphi(x_j) = y_i y_j \, K(x_i, x_j). \tag{11}$$

For the kernel function $K(\cdot,\cdot)$ one typically has the following choices:

$$\begin{array}{ll} K(x,x_i) = x_i^T x, & \text{(linear kernel)} \\ K(x,x_i) = (1+x_i^T x/c)^d, & \text{(polynomial kernel of degree } d) \\ K(x,x_i) = \exp\{-\|x-x_i\|_2^2/\sigma^2\}, & \text{(RBF kernel)} \\ K(x,x_i) = \tanh(\kappa\, x_i^T x + \theta), & \text{(MLP kernel)}, \end{array}$$

where d, c, σ, κ and θ are constants. Notice that the Mercer condition holds for all $c, \sigma \in \mathbb{R}^+$ and $d \in \mathbb{N}$ values in the polynomial and RBF case, but not for all possible choices of κ and θ in the MLP case. The scale parameters c, σ and κ determine the scaling of the inputs in the polynomial, RBF and MLP kernel function. This scaling is related to the bandwidth of the kernel in statistics, where it is shown that the bandwidth is an important parameter of the generalization behavior of a kernel method [34]. The LS-SVM classifier is then constructed as follows:

$$y(x) = \text{sign}[\sum_{i=1}^{N} \alpha_i y_i K(x,x_i) + b]. \tag{12}$$

A simple and practical way to construct binary MLP classifiers is to use a regression formulation [5, 16], where one uses the targets ± 1 to encode the first and second class, respectively. By the use of equality constraints with targets $\{-1,+1\}$, the LS-SVM formulation (6)-(7) implicitly corresponds to a regression formulation with regularization [5, 16, 36, 44]. Indeed, by multiplying the error e_i with $y_i \in \{-1,+1\}$, the error term $E_D = \sum_{i=1}^{N} e_i^2$ becomes

$$E_D = \frac{1}{2}\sum_{i=1}^{N} e_i^2 = \frac{1}{2}\sum_{i=1}^{N}(y_i e_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(y_i - (w^T\varphi(x_i)+b))^2. \tag{13}$$

Neglecting the bias term in the LS-SVM formulation, this regression interpretation relates the LS-SVM classifier formulation to regularization networks [17, 40], Gaussian Processes regression [58] and to the ridge regression type of SVMs for nonlinear function estimation [36, 41]. SVMs and LS-SVMs give the additional insight of the kernel-induced feature space, while the same expressions are obtained in the dual space as with regularization networks for function approximation and Gaussian Processes on the first level of the evidence framework [29, 53].

Given this regression interpretation, the bias term in the LS-SVM formulation allows to explicitly relate the LS-SVM classifier to regularized Fisher's linear discriminant analysis in the feature space (ridge regression) [19]. Fisher's linear discriminant is defined as the linear function with maximum ratio of between-class scatter to within-class scatter [5, 16, 18]. By defining N_+ and N_- as the number of training data of class +1 and −1, respectively, Fisher's linear discriminant (in the feature space) with regularization is obtained by minimizing

$$\min_{w_F, b_F} \frac{1}{2} w_F^T w_F + \frac{\gamma_F}{2} \sum_{i=1}^{N} (t_i - w_F^T \varphi(x_i) + b_F)^2, \qquad (14)$$

with appropriate targets $t_i = -(N/N_-)$ if $y_i = -1$ and $t_i = N/N_+$ if $y_i = +1$ [18].

This least squares regression problem [5, 16] yields the same linear discriminant w_F as is obtained from a generalized eigenvalue problem in [3, 30], where the bias term is determined by, e.g., cross-validation. The regression formulation (14) with Fisher's targets $\{-N/N_-, +N/N_+\}$ chooses the bias term b_F based on the sample mean [5, 16], while the targets $\{-1, +1\}$ correspond to an asymptotically optimal least squares approximation of Bayes' discriminant for two Gaussian distributions [16]. The difference between the LS-SVM classifier and the regularized Fisher discriminant is that the corresponding regression interpretations use different targets being $\{-1, +1\}$ and $\{-N/N_-, +N/N_+\}$, respectively. It can be proven that for $\gamma_F = \gamma$, the relation between the two classifier formulations is given $2w_F = w$ and $2b_F = b - \frac{1}{N}\sum_{i=1}^{N} y_i$. Hence, by solving the linear set of equations (10) in the dual space for the LS-SVM classifier, e.g., by a large scale algorithm [21, 45], the regularized Fisher discriminant is also obtained. Both choices for the targets will be compared in the experimental setup, where the Fisher discriminant is denoted by LS-SVM$_\mathrm{F}$ (LS-SVM with Fisher's discriminant targets).

3. Multiclass LS-SVMs

Multiclass categorization problems are typically solved by reformulating the multiclass problem with M classes into a set of L binary classification problems [2, 5, 35, 51]. To each class $\mathcal{C}_m$, $m = 1, \ldots, M$, a unique codeword $c_m = [y_m^{(1)}, y_m^{(2)}, \ldots, y_m^{(L)}] = y_m^{(1:L)} \in \{-1, 0, +1\}^L$ is assigned, where each binary classifier $f_l(x)$, $l = 1, \ldots, L$, discriminates between the corresponding output bit y_l. The use of 0 in the codeword is explained below.

There exist different approaches to construct the set of binary classifiers. In classical neural networks different outputs are defined to encode up to multiple classes. One uses L outputs in [46] to encode up to 2^L classes. This output coding, having minimal L, is referred to as minimum output coding (MOC). Other output codes have been proposed in the literature [5, 51]. In one-versus-all (1vsA) coding with $M = L$ one makes binary decisions between one class and all other classes [5, 29]. Error correcting output codes (ECOC) [15] are motivated by information theory and introduce redundancy ($M < L$) in the output coding to handle misclassifications of the binary classifier. In one-versus-one (1vs1) output coding [23], one uses $M(M-1)/2$ binary plug-in classifiers, where each binary classifier discriminates between two opposing classes. The 1vs1 output coding can also be represented by L-bit codewords $c_m \in \{-1, 0, +1\}^L$ when one uses a 0 for the classes that are not considered. For example, three classes can be represented using the codewords $c_1 = [-1; +1; 0]$, $c_2 = [-1; 0; +1]$ and $c_3 = [0; -1; +1]$. The MOC and 1vs1 coding are illustrated for an artificial three class problem in Figure 1, respectively. While MOC uses only 2 bits ($L = 2$) to encode 3 classes, the use of three output bits in the 1vs1 coding typically results into simpler decision boundaries.

The L classifiers each assign an output bit $y^{(l)} = \text{sign}[f^{(l)}(x)]$ to a new input vector x. The class label is typically assigned [2, 15, 39] to the corresponding output code with minimal Hamming distance $\Delta_H(y^{(1:L)}, c_m)$, with

$$\Delta_H(y^{(1:L)}, c_m) = \sum_{l=1}^{L} \begin{cases} 0 & \text{if } y^{(l)} = y_m^{(l)} \text{ and } y^{(l)} \neq 0 \text{ and } y_m^{(l)} \neq 0 \\ \frac{1}{2} & \text{if } y^{(l)} = 0 \text{ or } y_m^{(l)} = 0 \\ 1 & \text{if } y^{(l)} \neq y_m^{(l)} \text{ and } y^{(l)} \neq 0 \text{ and } y_m^{(l)} \neq 0. \end{cases}$$

When one considers the 1vs1 output coding scheme as a voting between each pair of classes, it is easy to see that the codeword with minimal Hamming distance corresponds to the class with the maximal number of votes.

In this paper, we restrict ourselves to the use of minimum output coding (MOC) [46] and one versus one (1vs1) coding [23]. Each binary classifier $f^{(l)}(x)$, $l = 1, \ldots, L$, is inferred on the training set $D^{(l)} = \{(x_i, y_i^{(l)}) | i = 1, \ldots, N \text{ and } y_i^{(l)} \in \{-1, +1\}\}$, consisting of $N^{(l)} \leq N$ training points, by solving

$$\left[\begin{array}{c|c} 0 & y^{(l)T} \\ \hline y^{(l)} & \Omega^{(l)} + {\gamma^{(l)}}^{-1} I \end{array}\right] \left[\begin{array}{c} b^{(l)} \\ \hline \alpha^{(l)} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 1_v \end{array}\right] \tag{15}$$

where $\Omega_{ij,l} = K_l(x_i, x_j)$. The binary classifier $f_l(x)$ is then obtained as $f^{(l)}(x) = \text{sign}[\sum_{i=1}^{N^{(l)}} y_i^{(l)} \alpha_i^{(l)} K^{(l)}(x, x_i) + b^{(l)}]$.

Since the binary classifiers $f^{(l)}(x)$ are trained independently, the superscript (l) will be omitted in the next Sections in order to simplify the notation.

4. Sparse approximation using LS-SVMs

As mentioned before, a drawback of LS-SVMs in comparison with standard QP type SVMs is that sparseness is lost due to the choice of the 2-norm in (6), which is also clear from the fact that $\alpha_i = \gamma e_i$ in (9). For standard SVMs one typically has that many α_i values are exactly equal to zero. In [47], it was shown how sparseness can be imposed to LS-SVMs by a pruning procedure which is based upon the sorted support value spectrum. This is important considering the equivalence between SVMs and sparse approximation, shown in [20]. Indeed, the α_i values obtained from the linear system (10) reveal the relative importance of the training data points with respect to each other. This information is then employed to remove less important points from the training set, where the omitted data points correspond to zero α_i values. An important difference with pruning methods in classical neural networks [5, 22, 27], e.g., optimal brain damage and optimal brain surgeon, is that in the LS-SVM pruning procedure no inverse of a Hessian matrix has to be computed. The LS-SVM pruning procedure can also be related to Interior Point and IRWLS methods for SVMs [31, 41], where a linear system of the same form as (10) is solved in each iteration step until the conditions for optimality and the resulting sparseness property of the SVM are obtained. In each step of the IRWLS solution the whole training set is still taken into account and the sparse SVM solution is obtained after convergence. The LS-SVM pruning procedure removes a certain percentage of training data points in each iteration step. The pruning of large positive and small negative α_i values results into support vectors that are located far from and near to the decision boundary. An LS-SVM pruning procedure in which only the support values near the decision boundary with large α_i does yield a poorer generalization behavior, which can be intuitively understood since the LS-SVMs in the last steps are trained only on a specific part of the global training set. In this sense, the sparse LS-SVM is somewhat in between the SVM solution with support vectors near the decision boundary and the relevance vector machine [50] with support vectors far from the decision boundary. When one assumes that the variance of the noise is not constant or when the dataset may contain outliers, one can also use a weighted least squares cost function [47, 52]. In this case sparseness is also introduced by putting the weights in the cost function to zero for data points with large errors.

Hence, by plotting the spectrum of the sorted $|\alpha_i|$ values of the binary LS-SVM, one can evaluate which data points are most significant for contributing to the LS-SVM classifier (12). Sparseness is imposed in a second stage by gradually omitting the least important data from the training set using a pruning procedure [47]: in each pruning step all data points of which the absolute value of the support value is smaller than a threshold are removed. The height of the threshold is chosen such that in each step, e.g., 5% of the total number of training points are removed. The LS-SVM is then re-estimated on the reduced training set and the pruning procedure is repeated until a user-defined performance index starts decreasing. The pruning procedure consists of the following steps:

1. Train LS-SVM based on N points.

2. For each output, remove a small amount of points (e.g., 5% of the set) with smallest values in the sorted $|\alpha_i|$ spectra.

3. Re-train the LS-SVM based on the reduced training set [44]. In order to increase a user-defined performance index and to be able to prune more, one can refine the hyper- and kernel parameters.

4. Go to 2, unless the user-defined performance index degrades significantly. A one-tailed paired t-test can be used, e.g., in combination with 10-fold cross-validation to report significant decreases in the average validation performance.

For the multiclass case, this pruning procedure is applied to each binary classifier $f^{(l)}(x)$, $l = 1, \ldots, L$.

5. Implementation

In this Section, an iterative implementation is discussed for solving the linear system (10). Efficient iterative algorithms, such as Krylov subspace and Conjugate Gradient (CG) methods, exists in numerical linear algebra [21] to solve a linear system $\mathcal{A}x = \mathcal{B}$ with positive definite matrix $\mathcal{A} = \mathcal{A}^T > 0$. Considering the cost function $V(x) = \frac{1}{2}x^T\mathcal{A}x - x^T\mathcal{B}$, the solution of the corresponding linear system is also found as $\arg\min_x V(x)$. In the Hestenes-Stiefel conjugate gradient algorithm [21], one starts from an initial guess x_0 and $V(x_i)$ is decreased in each iteration step i as follows:

$i = 0; x = x_0; r = \mathcal{B} - \mathcal{A}x_i; \rho_0 = ||r||_2^2$
while $(i < i_{max}) \wedge (\sqrt{\rho_i} > \epsilon_1 ||\mathcal{B}||_2) \wedge (V(x_{i-1}) - V(x_i) > \epsilon_2)$,
  $i = i + 1$
  if $i = 1$
    $p = r$
  else
    $\beta_i = \rho_{i-1}/\rho_{i-2}$
    $p = r + \beta_i p$
  endif
  $w = \mathcal{A}p$
  $\alpha_i = \rho_{i-1}/(p^T w)$
  $x = x + \alpha_i$
  $r = r - \alpha_i w$
  $\rho_i = ||r||_2^2$
  $V(x_i) = -\frac{1}{2} x_i^T (r + \mathcal{B})$
endwhile

For $\mathcal{A} \in \mathrm{IR}^{N \times N}$ and $\mathcal{B} \in \mathrm{IR}^N$, the algorithm requires one matrix-vector multiplication $w = \mathcal{A}p$ and $10N$ flops, while only four vectors of length N are stored: x, r, p and w. For $\mathcal{A} = I + \mathcal{C} \geq 0$ and $\mathrm{rank}(\mathcal{C}) = r_c$, the algorithm converges in at most $r_c + 1$ steps, assuming infinite machine precision. However, the rate of convergence may be much higher, depending on the condition number of $\mathcal{A}$. The algorithm stops when one of the three stopping criteria is satisfied. While the first criterion stops the algorithm after maximal i_{max} iteration steps, the second criterion is based on the norm of the residuals. The third criterion is based on the evolution of the cost function $V(x)$, which is evaluated as $V(x) = \frac{1}{2} x^T (\mathcal{A}x - \mathcal{B}) - \frac{1}{2} x^T \mathcal{B} = -\frac{1}{2} x^T (r + \mathcal{B})$. The constants i_{max}, ϵ_1 and ϵ_2 are determined by the user, e.g., according to the required numerical accuracy. The use of iterative algorithms to solve large scale linear systems is preferred over the use of direct methods, since direct methods would involve a computational complexity of $O(N^3)$ and memory requirements of $O(N^2)$ [21], while the computational complexity of the CG algorithm is at most $O(r_c N^2)$ when $\mathcal{A}$ is stored. As the number of iterations i is typically smaller than r_c, an additional reduction in the computational requirements is obtained. When the matrix $\mathcal{A}$ is too large for the memory requirements, one can recompute $\mathcal{A}$ in each iteration step, which costs $O(N^2)$ operations per step but also reduces the memory requirements to $O(N)$.

In order to apply the CG algorithm to (10), the system matrix $\mathcal{A}$ involved in the set of linear equations should be positive definite.

Therefore, according to [45] the system (10) is transformed into

$$\begin{bmatrix} s & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} b \\ \alpha + H^{-1} Y b \end{bmatrix} = \begin{bmatrix} -0 + y^T H^{-1} 1_v \\ 1_v \end{bmatrix} \tag{16}$$

with $s = Y^T H^{-1} Y > 0$ and $H = H^T = \Omega + \gamma^{-1} I_N > 0$. The system (10) is now solved as follows [45]:

1. Use the CG algorithm to solve η, ν from

$$H\eta = y \tag{17}$$

$$H\nu = 1_v. \tag{18}$$

2. Compute $s = y^T \eta$.

3. Find solution: $b = \eta^T 1_v / s$ and $\alpha = \nu - \eta b$.

When no information on the solution x is available, one typically chooses $x_0 = 0$ for solving (17) and (18). In the next Section, the initial guess will be used to speed up the calculations when solving (10) for different choices of the hyperparameter γ. For a new γ_{new} value, the initial guess for η and ν can be based on the assumption that the training set errors $e_{i,\text{new}}$ do not significantly differ from the errors $e_{i,\text{old}}$, $i = 1, \ldots, N$ corresponding to the previous choice γ_{old} [41]. Hence we can write in vector notation $e_{\text{new}} \simeq e_{\text{old}}$ or $\alpha_{\text{new}} \simeq \gamma_{\text{new}}/\gamma_{\text{old}} \alpha_{\text{old}}$, while the bias term is not changed $b_{\text{new}} \simeq b_{\text{old}}$. From (17) and (18) in step 1 and from the solutions for α and b in steps 2 and 3, it can be seen that these initial guesses for α_{new} and b_{new} are obtained by choosing $\eta_{new,0} = \gamma_{\text{new}}/\gamma_{\text{old}} \eta_{\text{old}}$ and $\nu_{new,0} = \gamma_{\text{new}}/\gamma_{\text{old}} \nu_{\text{old}}$. Krylov subspace methods also allow to solve the linear system simultaneously for different regularization parameters γ.

For large N, the matrix $\mathcal{A} = H = \Omega + \gamma^{-1} I_N$ with dimensions $N \times N$ cannot be stored due to memory limitations, the elements $\mathcal{A}_{ij}$ have to be re-calculated in each iteration. Since $\mathcal{A}$ is symmetric, this requires $N(N-1)/2$ kernel function evaluations. Since $\mathcal{A}$ is equal in (17) and (18), the number of kernel function evaluations is reduced by a factor 1/2 when simultaneously solving (17) and (18) for η and ν. Also observe that the condition number increases when γ is increased or when less weight is put onto the regularization term. We used at most $i_{max} = 150$ iterations and put the other constants to $\epsilon_1 = \epsilon_2 = 10^{-9}$, which is giving good results on all tried datasets.

6. Hyperparameter selection

Different techniques exist for tuning the hyperparameters related to the regularization constant and the parameter of the kernel function. Among the available tuning methods we find minimization of the VC-dimension [5, 41, 43, 54, 55], the use of cross-validation methods, bootstrapping techniques, Bayesian inference [5, 26, 29, 52, 53], etc. In this Section, the regularization and kernel parameters of each binary classifier are selected using a simple 10-fold cross-validation procedure.

In the case of an RBF kernel, the hyperparameter γ, the kernel parameter σ and the test set performance of the binary LS-SVM classifier are estimated using the following steps:

1. Set aside 2/3 of the data for the training/validation set and the remaining 1/3 for testing.

2. Starting from $i = 0$, perform 10-fold cross-validation on the training/validation data for each (σ, γ) combination from the initial candidate tuning sets[1] $\Sigma_0 = \{0.5, 5, 10, 15, 25, 50, 100, 250, 500\} \cdot \sqrt{n}$ and $\Gamma_0 = \{0.01, 0.05, 0.1, 0.5,\ 1, 5, 10, 50, 100, 500, 1000\}$.

3. Choose optimal (σ, γ) from the tuning sets Σ_i and Γ_i by looking at the best cross-validation performance for each (σ, γ) combination.

4. If $i = i_{max}$, go to step 5; else $i := i + 1$, construct a locally refined grid $\Sigma_i \times \Gamma_i$ around the optimal hyperparameters (σ, γ) and go to step 3.

5. Construct the LS-SVM classifier using the total training/validation set for the optimal choice of the tuned hyperparameters (σ, γ).

6. Assess the test set accuracy by means of the independent test set.

In this benchmark study, i_{max} was typically choosen to $i_{max} = 3$ using 2 additional refine searches. This involves a fine-tuned selection of the σ and γ parameters. It should be remarked that the refining of the grid is not always necessary as the 10-fold (CV10) cross-validation performance typically has a flat maximum, as can be seen from Figure 2.

In combination with the iterative algorithm of Section 5, one solves the linear systems (17)-(18) for the first grid point starting from initial value $x_0 = 0$. For the next γ value in the grid, the solutions η and ν were initialized accordingly. When changing σ, we initialized ν and η with

[1] The square root $\sqrt{n}$ of the number of inputs n is considered in the grid Σ (Step 2) since $||x - x_i||_2^2$ in the RBF kernel is proportional to n.

the corresponding solutions related to the previous σ value. Depending on the distance between the points in the grid, the average reduction of the number of iteration steps amounts to 10%-50% with respect to starting from $x_0 = 0$ for all (σ, γ) combinations.

For the polynomial kernel functions the hyperparameters γ and c were tuned by a similar procedure, while γ parameter of the linear kernel was selected from a refined set Γ based upon the cross-validation performance. For multiclass problems, the cross-validation procedure is repeated for each binary classifier $f^{(l)}(x)$, $l = 1, \ldots, L$.

7. Benchmark Results

In this Section, we report on the application of LS-SVMs on 20 benchmark datasets [6], of which a brief description will be included in Subsection 7.1. The performance of the LS-SVM classifier is compared with a selection of reference techniques discussed in Subsection 7.2. In Subsections 7.3 and 7.4 the randomized test set results are discussed. The sparse approximation procedure is illustrated in Subsection 7.5.

7.1. Description of the datasets

Most datasets have been obtained from the publicly accessible UCI benchmark repository [6] at `http://kdd.ics.uci.edu/`. The US postal service dataset was retrieved from the Kernel Machines website at `http://www.kernel-machines.org/`. These datasets have been referred to numerous times in the literature, which makes them very suitable for benchmarking purposes. As a preprocessing step, all records containing unknown values are removed from consideration. The following binary datasets were retrieved from [6]: the Statlog Australian credit (`acr`), the Bupa liver disorders (`bld`), the Statlog German credit (`gcr`), the Statlog heart disease (`hea`), the Johns Hopkins university ionosphere (`ion`), the Pima Indians diabetes (`pid`), the sonar (`snr`), the tic-tac-toe endgame (`ttt`), the Wisconsin breast cancer (`wbc`) and the adult (`adu`) dataset. The main characteristics of these datasets are summarized in Table I. The following multiclass datasets were used: the balance scale (`bal`), the contraceptive method choice (`cmc`), the image segmentation (`ims`), the iris (`iri`), the LED display (`led`), the thyroid disease (`thy`), the US postal service (`usp`), the Statlog vehicle silhouette (`veh`), the waveform (`wav`) and the wine recognition (`win`) dataset. The main characteristics of the multiclass datasets are summarized in Table II.

7.2. Description of the reference algorithms

The test set performance of the LS-SVM classifier was compared with the performance of a selection of reference techniques[2]: the SVM classifier with linear and RBF kernel; the decision tree algorithm C4.5 [33], Holte's one-rule classifier (oneR) [24]; statistical algorithms like linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic regression (logit) [5, 16, 35]; instance based learners (IB) [1] and Naive Bayes [25]. The oneR, LDA, QDA, logit, NBk and NBn require no parameter tuning. For C4.5, we used the default confidence level of 25% for pruning, which is the value that is commonly used in the machine learning literature. We also experimented with other pruning levels on some of the datasets, but found no significant performance increases. For IB we used both 1 (IB1) and 10 (IB10) nearest neighbours. We used both standard Naive Bayes with the normal approximation ($\mathrm{NB_n}$) [16] and the kernel approximation ($\mathrm{NB_k}$) for numerical attributes [25]. The default classifier or majority rule (Maj. Rule) was included as a baseline in the comparison tables. All comparisons were made on the same randomizations. For another comparison study among 22 decision tree, 9 statistical and 2 neural network algorithms, we refer to [28].

The comparison is performed on an out-of-sample test set consisting of 1/3 of the data. The first 2/3 of the randomized data was reserved for training and/or cross-validation. For each algorithm, we report the average test set performance and sample standard deviation on 10 randomizations in each domain [4, 12, 13, 28]. The best average test set performance was underlined and denoted in bold face for each domain. Performances that are not significantly different at the 5% level from the top performance with respect to a one-tailed paired t-test are tabulated in bold face. Statistically significant underperformances at the 1% level are emphasized. Performances significantly different at the 5% level but not a the 1% level are reported in normal script. Since the observations on the randomizations are not independent [14], we remark that this standard t-test is used as a (common) heuristic to show that the average accuracies on the ten randomizations can be considered to be different. Ranks are assigned to each algorithm starting from 1 for the best average performance and ending with 18 and 28 for the algorithm with worst performance, for the binary and multiclass domains, respectively. Averaging over all domains, the Average Accuracy (AA) and Average Rank (AR) are reported for each

[2] The matlab SVM toolbox [10] with SMO solver [32] was used to train and evaluate the Vapnik SVM classifier. The C4.5, IB1, IB10, Naive Bayes and oneR algorithms were implemented using the Weka workbench [59], while the Discriminant Analysis Toolbox (M. Kiefte) for Matlab was applied for LDA, QDA and logit.

algorithm [28]. A Wilcoxon signed rank test of equality of medians is used on both AA and AR to check whether the performance of an algorithm is significantly different from the algorithm with the highest accuracy. A Probability of a Sign Test (P_{ST}) is also reported comparing each algorithm to the algorithm with best accuracy. The results of these significance tests on the average domain performances are denoted in the same way as the performances on each individual domain.

7.3. Performance of the binary LS-SVM classifier

In this Subsection, we present and discuss the results obtained by applying the empirical setup, outlined in Section 6, on the 10 UCI binary benchmark datasets described above. All experiments were carried out on Sun Ultra5 Workstations and on Pentium II and III PCs. For the kernel types, we used RBF kernels, linear (Lin) and polynomial (Pol) kernels (with degree $d = 2, \ldots, 10$). Both the performance of LS-SVM targets $\{-1, +1\}$ and Regularized Kernel Fisher Discriminant Analysis (LS-SVM$_\mathrm{F}$) targets $\{-N/N_-, +N/N_+\}$ are reported. All given n inputs are normalized to zero mean and unit variance [5].

The regularization parameter γ and the kernel parameters σ and c of the binary LS-SVM classifier with linear, RBF and polynomial kernel were selected from the 10-fold cross-validation procedure discussed in Section 6. The optimal values for these parameters are reported in Table III. The flat maximum of the CV10 classification accuracy is illustrated in Figure 2 for the `ion` dataset. This pattern was commonly encountered among all evaluated datasets. The corresponding training, validation and test set performances are reported in Tables V, VI and VII, respectively. The best validation and test set performances are underlined and denoted in bold face. These experimental results indicate that the RBF kernel yields the best validation and test set performance, while also polynomial kernels yield good performances. Note that we also conducted the analyses using non-scaled polynomial kernels, i.e., with $c = 1$. For this scaling parameter LS-SVMs with polynomial kernels of degrees $d = 2$ and $d = 10$ yielded on all domains average test set performances of 84.3% and 65.9%, respectively. Comparing this performance with the average test set performance of 85.6% and 85.5% (Table VII) obtained when using scaling, this clearly motivates the use of bandwidth or kernel parameters. This is especially important for polynomial kernels with degree $d \geq 5$.

The regularization parameter C and kernel parameter σ of the SVM classifiers with linear and RBF kernels were selected in a similar way as for the LS-SVM classifier using the 10-fold cross-validation procedure. The optimal hyperparameters of the SVM classifiers were reported in

Table IV. The corresponding average test set performances are given in Table VIII.

The optimal regularization and kernel parameters were then used to assess the test set performance of the LS-SVM and LS-SVM$_{\mathrm{F}}$ classifier on 10 randomizations: for each randomization the first 2/3 of the data were used for training, while the remaining 1/3 was put aside for testing. In the same way, the test set performances of the reference algorithms were assessed. The same randomizations were used to tabulate the performances of the reference algorithms. Both sample mean and sample standard deviation of the performance on the different domains are denoted in Table VIII using the bold, normal and emphasized script to enhance the visual interpretation as explained above. Averaging over all domains, the mean performance and rank and the probability of different medians with respect to the best algorithm are tabulated in the last 3 columns of the Table.

The LS-SVM classifier with Radial Basis Function kernel (RBF LS-SVM) achieves the best average test set performance on 3 of the 10 benchmark domains, while its accuracy is not significantly worse than the best algorithm in 3 other domains. LS-SVM classifiers with polynomial and linear kernel yield the best performance on two and one datasets, respectively. Also RBF SVM, IB1, NB$_{\mathrm{k}}$ and C4.5 achieve the best performance on one dataset each. Comparison of the accuracy achieved by the nonlinear polynomial and RBF kernel with the accuracy of the linear kernel illustrates that most domains are only weakly nonlinear. The LS-SVM formulation with targets $\{-1,+1\}$ yields a better performance than the LS-SVM$_{\mathrm{F}}$ regression formulation related to regularized kernel Fisher's discriminant with targets $\{-N/N_-,+N/N_+\}$, although not all tests report a significant difference. Noticing that the LS-SVM with linear kernel and without regularization ($\gamma \to \infty$) corresponds to the LDA classifier, we also remark that a comparison of both accuracies indicates that the use of regularization slightly improves the generalization behaviour.

Considering the Average Accuracy (AA) and Average Ranking (AR) over all domains [4, 12, 13], the RBF SVM gets the best average accuracy and the RBF LS-SVM yields the best average rank. There is no significant difference between the performance of both classifiers. The average performance of Pol LS-SVM and Pol LS-SVM$_{\mathrm{F}}$ is not significantly different with respect to the best algorithms. The performances of many other advanced SVM algorithms are in line with the above results [8, 30, 38]. The significance tests on the average performances of the other classifiers do not always yield the same results. Generally speaking, the performance of Lin LS-SVM, Lin SVM, Logit, NB$_{\mathrm{k}}$ and IB1 are not significantly different at the 1% level. Also the performances of LS-SVMs with Fisher's discriminant targets (LS-SVM$_{\mathrm{F}}$) are not signifantly different at the 1%. These results allow to conclude that the SVM and LS-SVM formulations achieve very good test set performances compared to the other reference classification algorithms.

7.4. Performance of the multiclass LS-SVM classifier

We report the performance of multiclass LS-SVMs on 10 multiclass categorization problems. Each multiclass problem is decomposed into a set of binary classification problems using minimum output coding (MOC) and one-versus-one (1vs1) output coding. The same kernel types as for the binary domain were considered: RBF kernels, linear (Lin) and polynomial (Pol) kernels with degrees $d = 2, \ldots, 10$. Both the performance of LS-SVM and LS-SVM$_{\mathrm{F}}$ classifiers are reported. The MOC and 1vs1 output coding were also applied to SVM classifiers with linear and RBF kernels. As for the binary domains, we normalized the inputs to zero mean and unit variance [5].

The experimental setup of the previous Subsection is used: each binary classifier of the multiclass LS-SVM is designed on the first 2/3 of the data using 10-fold cross-validation, while the remaining 1/3 are put aside for testing. The selected regularization and kernel parameters[3] were then fixed and 10 randomizations were conducted on each domain. The average test set accuracies of the different LS-SVM and LS-SVM$_{\mathrm{F}}$ classifiers, with RBF, Lin and Pol kernel ($d = 2, \ldots, 10$) and using MOC and 1vs1 output coding, are reported in Table IX. The test set accuracies of the reference algorithms on the same randomizations are also reported, where we remark that for the `usp` dataset the memory requirements for logit were too high. Instead, we tabulated the performance of LDA instead of logit for this single case. The same statistical tests as in Subsection 7.3 were used to compare the performance of the different classifiers.

The use of QDA yields the best average test set accuracy on two domains, while LS-SVMs with 1vs1 coding using a RBF and Lin kernel and LS-SVM$_{\mathrm{F}}$ with Lin kernel each yield the best performance on one domain. SVMs with RBF kernel with MOC and 1vs1 coding yield the best performance on one domain each. Also C4.5, logit and IB1 each achieve one time the best performance. The use of 1vs1 coding generally results into a better classification accuracy. Averaging over all 10 multiclass domains, the LS-SVM classifier with RBF kernel and 1vs1 output coding achieves the best average accuracy (AA) and average ranking, while its performance is only on three domains significantly worse at 1%

[3] The corresponding optimal values for the hyperparameters are not reported, because of the large number of binary classifiers for some datasets.

than the best algorithm. This performance is not significantly different from the SVM with RBF kernel and 1vs1 output coding. Summarizing the different significant tests, RBF LS-SVM (MOC), Pol LS-SVM (MOC), Lin LS-SVM (1vs1), Lin LS-SVM$_{\mathrm{F}}$ (1vs1), Pol LS-SVM (1vs1), RBF SVM (MOC), RBF SVM (1vs1), Lin SVM (1vs1), LDA, QDA, C4.5, IB1 and IB10 perform not significantly different at the 5% level. While NB$_{\mathrm{k}}$ performed well on the binary domains, its average accuracy on the multiclass domains is never comparable at the 5% level for all three tests. The results of Table IX illustrate that the SVM and LS-SVM classifier with RBF kernel using 1vs1 output coding consistently yield very good test set accuracies on the multiclass domains.

7.5. LS-SVM sparse approximation procedure

The sparseness property of SVMs is lost in LS-SVMs by the use of a 2-norm. While the generalization capacity is still controlled by the regularization term, the use of a smaller number of support vectors may be interesting to reduce the computational cost of evaluating the classifier for new inputs x. We illustrate the sparse approximation procedure [44, 46, 47] of Section 4 on 10 datasets for LS-SVMs with RBF kernel.

The amount of pruning was estimated by means of 10-fold cross-validation. In each pruning step, 5% percent of the support values of each training set were pruned and the classification accuracy was assessed on the corresponding validation set. The hyperparameters were re-estimated on a small local grid when the cross-validation performance decreased. This local optimization of the hyperparameters is carried out in order to prune more support values at the expense of an increased computational cost. Then, the next pruning step was performed. This pruning procedure was stopped when the cross-validation performance decreased significantly using a paired one-tailed t-test compared with the initial performance (when no pruning was applied) or with the previous performance. Both 1% and 5% Significance Levels where used and are denoted by SL1% and SL5%, respectively. Given the number of Pruning Steps $\#\mathrm{PS}_{\mathrm{SL5\%}}$ and $\#\mathrm{PS}_{\mathrm{SL1\%}}$ and the optimal hyperparameters for each step from the cross-validation procedure, the pruning is now performed in the same way on the whole cross-validation set starting from N_{CV} support values down to $N_{\mathrm{SL5\%}}$ and $N_{\mathrm{SL1\%}}$ support values and the accuracies of the LS-SVM, LS-SVM$_{\mathrm{SL5\%}}$ and LS-SVM$_{\mathrm{SL1\%}}$ on the test set are reported. An example of the evolution of the cross-validation and test set performance for the `wbc` dataset as a function of the number of pruning steps is depicted in

Figure 3. For multiclass domains, 1vs1 coding was used and the sparse approximation procedure was applied to each binary classifier.

This pruning procedure was applied on the first 5 binary and first 5 multiclass problems studied above, using LS-SVMs with RBF kernels and with 1vs1 output coding for the multiclass problems. The resulting randomized test set accuracies of the LS-SVM$_{\mathrm{SL5\%}}$ and LS-SVM$_{\mathrm{SL1\%}}$ are tabulated in Table X, corresponding to pruning till SL5% and SL1% significance levels, respectively. The results of applying no pruning (LS-SVM) at all are repeated on the first row. A one-tailed paired t-test is used to highlight statistical differences from the best algorithm on each domain in the same way as in Tables VIII and IX. In most cases, the SL5% and SL1% stop criterion yield no significant difference in the number of pruning steps and the resulting number of support values, while the performance generally decreases not significantly. Experimentally we observe that the performance on a validation set stays constant or increases slightly when pruning. After a certain time, the performance starts decreasing rather fast as the pruning continues. This explains the small difference in the number of pruned training examples when using the 5% and 1% stopping criterion. Averaging over all datasets, we find that the sparse approximation procedure does not yield significantly different results at the 1% level. From a comparison with Tables VIII and IX, we conclude that this procedure allows a firm reduction of the number of training examples, while still a good test set performance is obtained.

8. Conclusions

Support vector machines for binary classification is an important new emerging methodology in the area of machine learning and neural networks. The kernel based representation of SVMs allows to formulate the classifier problem as a convex optimization problem, usually a quadratic programming problem. Moreover the model complexity (e.g., number of hidden units in the case of RBF kernel) follows as a solution to the QP problem where the number of hidden units is equal to the number of non-zero support vector vectors. In LS-SVMs a least squares cost function is used and the solution of the binary LS-SVM follows from solving a linear Karush-Kuhn-Tucker system instead of a QP problem. Keeping in mind interior point methods for solving the QP problem, an iterative procedure is needed there, where at each iteration step a linear system has to be solved which has the same complexity as of one single LS-SVM. This binary LS-SVM formulation is related to a simple regression approach to classification using binary targets. In this sense, it is also related to regularization networks and Gaussian Processes regression. The use of the bias term in the formulation and the primal-dual formulation also allow to relate the LS-SVM classifier to Fisher's discriminant analysis in the feature space, also known as Kernel Fisher Discriminant Analysis. Multiclass categorization problems are reformulated as a set of binary classification problems using, e.g., minimum output coding and one-versus-one output coding. We have shown on twenty UCI binary and multiclass benchmark datasets that the SVM and LS-SVM formulation in combination with standard cross-validation methods for hyperparameter selection are performing consistently very well on all tried datasets in comparison with many other methods. We have also verified that the lack of sparseness of LS-SVMs can be circumvented by applying sparse approximation in a second stage by pruning less important data from the sorted support value spectrum. In all experiments LS-SVMs with RBF kernel and cross-validation tuning of the two hyperparameters are performing well and at least as good as LS-SVMs with linear kernel which, in combination with the sparse approximation procedure, also offers the possibility for knowledge discovery in the datasets.

Acknowledgements

T. Van Gestel and J.A.K. Suykens are a Research Assistant and Postdoctoral Researcher with the FWO Flanders, respectively. S. Viaene is researcher at LIRIS with the KBC Insurance Research Chair. G. Dedene is a co-founder of the KBC Insurance Research Chair. This work was carried out at the ESAT laboratory, the Interdisciplinary Center of Neural Networks ICNN and the institute LIRIS of the K.U.Leuven and supported by grants and projects from the Flemish Gov.: (Research council KUL: Grants, GOA-Mefisto 666; FWO-Flanders: Grants, res. proj. (G.0262.97, G.0135.95, G.0115.01, G.0197.02, G.0407.02) and comm. (ICCoS and ANMMM); AWI: Bil. Int. Coll.; IWT: STWW Eureka SINOPSYS, IMPACT, FLiTe; STWW Genprom, GBOU McKnow; from the Belgian Fed. Gov. (Interuniv. Attr. Poles: IUAP-P4/02, P4/24; Program Dur. Dev.); from the Eur. Comm.: (TMR Netw. (Alapedes, Niconet); Science: ERNSI). The work is also carried out in the framework of the KBC Insurance Research Chair, set up in September 1997 as a pioneering cooperation between LIRIS and KBC Insurance.

References

1. Aha, D., & Kibler, D. (1991). Instance-based learning algorithms. *Machine Learning*, 6, 37-66.
2. Allwein, E.L., Schapire, R.E., & Singer, Y. (2000). Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1, 113-141.
3. Baudat, G., & Anouar, F. (2000) Generalized Discriminant Analysis Using a Kernel Approach. *Neural Computation,* 12, 2385-2404.
4. Bay, S.D. (1999). Nearest Neighbor Classification from Multiple Feature Subsets. *Intelligent Data Analysis*, 3, 191-209.
5. Bishop, C.M. (1995). *Neural networks for pattern recognition.* Oxford University Press.
6. Blake, C.L., & Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/˜mlearn/MLRepository.html]. Irvine, CA: University of California, Dept. of Information and Computer Science.
7. Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In: *Proc. of the Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, ACM, (pp, 144-152).
8. Bradley, P.S., & Mangasarian, O.L. (1998). Feature Selection via Concave Minimization and Support Vector Machines. In: J. Shavlik (ed.), *Machine Learning Proc. of the Fifteenth Int. Conf. (ICML'98)*, Morgan Kaufmann, San Francisco, California, (pp. 82-90).
9. Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Chapman and Hall, New York.
10. Cawley, G.C. (2000). MATLAB Support Vector Machine Toolbox (v0.54β). [http://theoval.sys.uea.ac.uk/∼gcc/svm/toolbox]. University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K.
11. Cristianini, N., & Shawe-Taylor, J. (2000) *An Introduction to Support Vector Machines.* Cambridge University Press.
12. De Groot, M.H. (1986). *Probability and Statistics* (Second Edition). Reading, MA: Addison-Wesley.
13. Domingos, P. (1996). Unifying Instance-Based and Rule-Based Induction. *Machine Learning*, 24, 141-168.
14. Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10, 1895-1924.
15. Dietterich, T.G., & Bakiri, G. (1995). Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2, 263-286.
16. Duda, R.O. & Hart, P.E. (1973). *Pattern Classification and Scene Analysis*, John Wiley, New York.
17. Evgeniou, T., Pontil, M., & Poggio, T. (2001) Regularization Networks and Support Vector Machines. *Advances in Computational Mathematics*, 13, 1-50.
18. Fisher, R.A. (1936). The Use of Multiple Measurements in Taxonomic Problems, *Annals of Eugenics*, 7(2), 179-188.
19. Friedman, J. (1989). Regularized Discriminant Analysis. *Journal of the American Statistical Association*, 84, 165-175.
20. Girosi, F. (1998). An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10, 1455-1480.
21. Golub, G.H., & Van Loan C.F. (1989). *Matrix Computations.* Baltimore MD: Johns Hopkins University Press.

22. Hassibi, B., & Stork, D.G. (1993). Second order derivatives for network pruning: optimal brain surgeon. In Hanson, Cowan, & Giles (Eds.), *Advances in Neural Information Processing Systems* (Vol.5). San Mateo, CA: Morgan Kaufmann.
23. Hastie, T., & Tibshirani, R. (1998). Classification by Pairwise Coupling, *The Annals of Statistics*, 26, 451–471.
24. Holte, R.C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11, 63-90.
25. John, G.H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, Quebec, Morgan Kaufmann, pp. 338-345.
26. Kwok, J.T. (2000). The evidence framework applied to Support Vector Machines. *IEEE Trans. on Neural Networks,* 10(5), 1018-1031.
27. Le Cun, Y., Denker, J.S., & Solla, S.A. (1990). Optimal brain damage. In Touretzky (Ed.), *Advances in Neural Information Processing Systems* (Vol.2). San Mateo, CA: Morgan Kaufmann.
28. Lim, T.-S., Loh, W.-Y., & Shih, Y.-S. (2000). A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms. *Machine Learning*, 40(3), 203-228.
29. MacKay, D.J.C. (1995). Probable Networks and Plausible Predictions - A Review of Practical Bayesian Methods for Supervised Neural Networks. *Network: Computation in Neural Systems*, 6, 469-505.
30. Mika, S., Rätsch, G., Weston, J., Schölkopf, B., & Müller, K.-R. (1999). Fisher Discriminant Analysis with Kernels. *Proc. IEEE Neural Networks for Signal Processing Workshop 1999, NNSP 99.*
31. Navia-Vázquez, A., Pérez-Cruz, F., Artés-Rodríguez, A., Figueiras-Vidál, A.R. (2001). Weighted least squares training of support vector classifiers leading to compact and adaptive schemes. *IEEE Transactions on Neural Networks,* 12, 1047 -1059.
32. Platt, J. (1998). Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C.J.C. Burges & A.J. Smola (Eds.), *Advances in Kernel Methods – Support Vector Learning.* Cambridge, MA.
33. Quinlan, J. (1993). *C4.5 Programs for Machine Learning.* Morgan Kaufmann.
34. Rao, P. (1983) *Nonparametric Functional Estimation*, Academic Press, Orlando.
35. Ripley, B.D. (1996). *Pattern Classification and Neural Networks.* Cambridge.
36. Saunders, C., Gammerman, A., & Vovk, V. (1998). Ridge Regression Learning Algorithm in Dual Variables. *Proc. of the 15th Int. Conf. on Machine Learning ICML-98*, Morgan Kaufmann, Madison-Wisconsin, (pp. 515-521).
37. Schölkopf, B., Sung, K.-K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., & Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing,* 45, 2758-2765.
38. Schölkopf, B., Burges, C., & Smola, A. (Eds.), (1998). *Advances in Kernel Methods - Support Vector Learning.* MIT Press.
39. Sejnowski, T.J., & Rosenberg, C.R. (1987). Parallel networks that learn to pronounce English text. *Journal of Complex Systems,* 1(1), 145-168.
40. Smola, A., Schölkopf, B., & Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, 11, 637-649.
41. Smola A. (1999). *Learning with Kernels.* PhD Thesis, published by: GMD, Birlinghoven.

42. Suykens, J.A.K., & Vandewalle, J. (Eds.), (1998). *Nonlinear Modeling: advanced black-box techniques.* Kluwer Academic Publishers, Boston.
43. Suykens, J.A.K., & Vandewalle, J. (1999). Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, 10, 907-912.
44. Suykens, J.A.K., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9, 293-300.
45. Suykens, J.A.K., Lukas, L., Van Dooren, P., De Moor, B., & Vandewalle, J. (1999). Least squares support vector machine classifiers: a large scale algorithm. *Proc. of the European Conf. on Circuit Theory and Design, (ECCTD'99)* (pp. 839-842).
46. Suykens, J.A.K., & Vandewalle, J. (1999). Multiclass Least Squares Support Vector Machines. *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN'99)*, Washington DC.
47. Suykens, J.A.K., De Brabanter, J., Lukas, L., Vandewalle J. (2001). Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, in press.
48. Suykens, J.A.K., & Vandewalle, J. (2000). Recurrent least squares support vector machines. *IEEE Transactions on Circuits and Systems-I*, 47, 1109-1114.
49. Suykens, J.A.K., Vandewalle, J., & De Moor, B. (2001). Optimal control by least squares support vector machines. *Neural Networks*, 14, 23-35.
50. Tipping, M.E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1, 211-244.
51. Utschick, W. (1998). A regularization method for non-trivial codes in polychotomous classification, *International Journal of Pattern Recognition and Artificial Intelligence*, 12, 453-474.
52. Van Gestel, T., Suykens, J.A.K., Baestaens, D.-E., Lambrechts, A., Lanckriet, G., Vandaele, B., De Moor, B., & Vandewalle, J. (2001). Predicting Financial Time Series using Least Squares Support Vector Machines within the Evidence Framework. *IEEE Transactions on Neural Networks, (Special Issue on Financial Engineering)*, 12, 809-821.
53. Van Gestel, T., Suykens, J.A.K., Lanckriet, G., Lambrechts, A., De Moor, B., & Vandewalle, J. (2001). A Bayesian Framework for Least Squares Support Vector Machine Classifiers, *Neural Computation*, in press.
54. Vapnik, V. (1995). *The nature of statistical learning theory.* Springer-Verlag, New-York.
55. Vapnik, V. (1998). *Statistical learning theory.* John Wiley, New-York.
56. Vapnik, V. (1998). The support vector method of function estimation. In Suykens, J.A.K., & Vandewalle, J. (Eds.), *Nonlinear Modeling: Advanced Black-box Techniques*, Kluwer Academic Publishers, Boston.
57. Viaene, S., Baesens, B., Van Gestel, T., Suykens, J.A.K., Van den Poel, D., Vanthienen, J., De Moor, B., & Dedene, G. (2001). Knowledge Discovery in a Direct Marketing Case using Least Squares Support Vector Machine Classifiers. *International Journal of Intelligent Systems*, 9, 1023-1036.
58. Williams, C.K.I. (1998). Prediction with Gaussian Processes: from Linear Regression to Linear Prediction and Beyond. In M.I. Jordan (Ed.), *Learning and Inference in Graphical Models.* Kluwer Academic Press.
59. Witten, I.H., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, San Francisco.

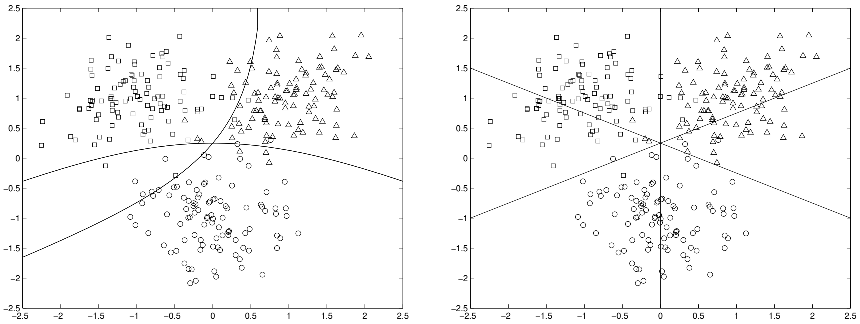

Figure 1. Importance of output coding on the shape of the binary decision lines illustrated on an artificial three class problem (∘, △ and □) with Gaussian distributions having equal covariance matrices. The shape of the binary decision lines depends on the representation of the multiclass problem problem as a set of binary classification problems. Left: minimal output coding (MOC) using 2 bits and assuming equal class distributions, the first binary classifier discriminates between (□, △) and ∘, while the second discriminates between □ and (△, ∘). Right: one-versus-one output (1vs1) coding using three bits, the three binary classifiers discriminate between □, △; □, ∘ and △, ∘, respectively. The use of MOC results into nonlinear decision lines, while linear decision lines are obtained with the 1vs1 coding.

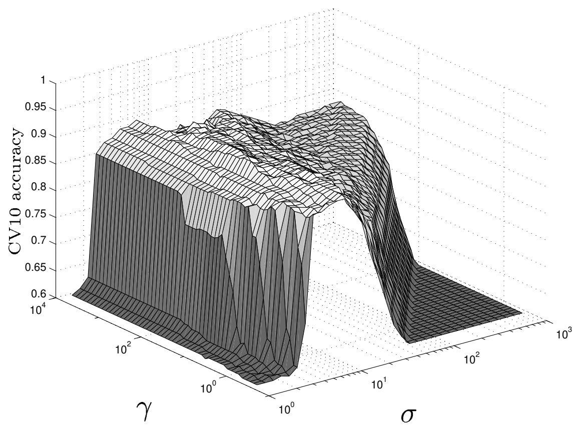

Figure 2. Cross-validation (CV10) classification accuracy on the `ion` dataset as a function of the regularization parameter γ and kernel parameter σ for an LS-SVM with RBF kernel. The function of this dataset is rather flat near the maximum. The CV10 accuracy is more sensitive to the kernel or bandwidth parameter σ selection [34] than to the choice of the regularization parameter for this dataset.

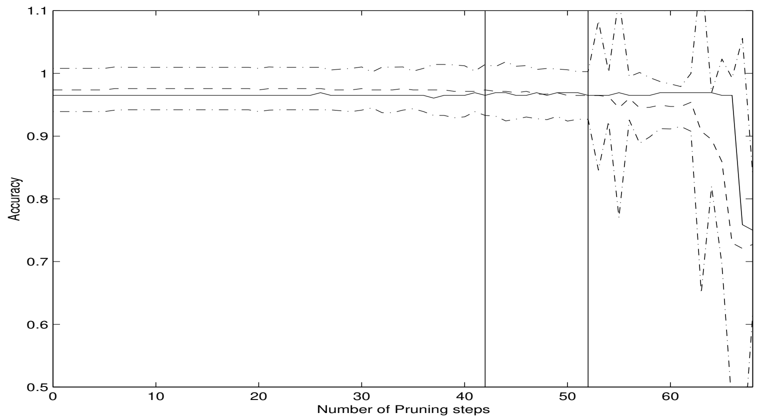

Figure 3. Illustration of the sparse approximation procedure with RBF LS-SVMs on the `wbc` dataset: evolution of the test set accuracy (full line) and cross-validation accuracy (dashed line) as a function of the number of pruning steps. The sample deviation on the cross-validation accuracy is also shown (dash-dotted line). In each step, 5% of the support values are pruned until the performance on the validation set starts to decrease significantly at the 5% and 1% level, respectively, marked by the vertical lines after $\#\mathrm{PS}_{\mathrm{SL}5\%} = 42$ and $\#\mathrm{PS}_{\mathrm{SL}1\%} = 52$ pruning steps. Starting from $N_{CV} = 455$ training examples, this number is reduced to $N_{\mathrm{SL}5\%} = 77$ and $N_{\mathrm{SL}1\%} = 58$ training examples corresponding to pruning percentages of $\mathrm{PPTE}_{\mathrm{SL}5\%}$=83% and $\mathrm{PPTE}_{\mathrm{SL}1\%}$=97%, respectively.

Table I. Characteristics of the binary classification UCI datasets, where N_{CV} stands for the number of data points used in the cross-validation based tuning procedure, N_{test} for the number of observations in the test set and N for the total dataset size. The number of numerical and categorical attributes is denoted by n_{num} and n_{cat} respectively, n is the total number of attributes.

	acr	bld	gcr	hea	ion	pid	snr	ttt	wbc	adu
N_{CV}	460	230	666	180	234	512	138	638	455	33000
N_{test}	230	115	334	90	117	256	70	320	228	12222
N	690	345	1000	270	351	768	208	958	683	45222
n_{num}	6	6	7	7	33	8	60	0	9	6
n_{cat}	8	0	13	6	0	0	0	9	0	8
n	14	6	20	13	33	8	60	9	9	14

Table II. Characteristics of the multiclass datasets, where N_{CV} stands for the number of data points used in the cross-validation based tuning procedure, N_{test} for the number of data in the test set and N for the total amount of data. The number of numerical and categorical attributes is denoted by n_{num} and n_{cat} respectively, n is the total number of attributes. The M row denotes the number of classes for each dataset, encoded by L_{MOC} and L_{1vs1} bits for MOC and 1vs1 output coding, respectively.

	bal	cmc	ims	iri	led	thy	usp	veh	wav	win
N_{CV}	416	982	1540	100	2000	4800	6000	564	2400	118
N_{test}	209	491	770	50	1000	2400	3298	282	1200	60
N	625	1473	2310	150	3000	7200	9298	846	3600	178
n_{num}	4	2	18	4	0	6	256	18	19	13
n_{cat}	0	7	0	0	7	15	0	0	0	0
n	4	9	18	4	7	21	256	18	19	13
M	3	3	7	3	10	3	10	4	3	3
L_{MOC}	2	2	3	2	4	2	4	2	2	2
L_{1vs1}	3	3	21	3	45	3	45	6	2	3

Table III. Optimized hyperparameter values of the LS-SVMs with RBF, linear and polynomial kernels for the binary classification datasets.

LS-SVM	acr	bld	gcr	hea	ion	pid	snr	ttt	wbc	adu
RBF: σ	22.75	41.25	31.25	5.69	3.30	240.00	33.00	2.93	6.97	10.0
RBF: $\log_{10}(\gamma)$	0.09	3.01	2.43	-0.76	0.63	3.04	0.86	2.20	-0.66	1.02
Lin: $\log_{10}(\gamma)$	-2.29	0.14	-1.82	-1.51	-1.99	-0.21	-2.26	-2.68	-1.53	-0.82
Pol: d	5	3	6	2	2	3	2	4	3	3
Pol: c	5.61	15.30	5.86	1.80	7.18	42.42	3.87	3.00	5.25	5.61
Pol: $\log_{10}(\gamma)$	-1.66	1.26	-1.20	-2.51	1.38	1.29	-1.27	0.45	-0.91	0.02

Table IV. Optimized hyperparameter values of the SVM with RBF and linear kernel for the binary classification datasets.

SVM	acr	bld	gcr	hea	ion	pid	snr	ttt	wbc	adu
RBF: σ	12.43	9.0	55.0	7.15	3.30	15.50	5.09	9.00	19.5	8.00
RBF: $\log_{10}(C)$	2.09	1.64	3.68	-0.51	0.51	0.04	1.70	-0.41	1.86	0.70
Lin: $\log_{10}(C)$	-2.43	1.57	1.45	1.32	1.20	-2.08	-1.05	-4.25	-2.12	-2.30

Table V. **Training set** performance of LS-SVMs on 10 binary domains.

LS-SVM	acr	bld	gcr	hea	ion	pid	snr	ttt	wbc	adu
N_{CV}	460	230	666	180	234	512	138	638	455	33000
n	14	6	20	13	33	8	60	9	9	14
RBF	0.86	0.75	0.83	0.86	1.00	0.78	0.94	1.00	0.98	0.85
Lin	0.87	0.69	0.75	0.85	0.90	0.78	0.88	0.66	0.96	0.82
Pol $d = 2$	0.89	0.75	0.82	0.87	0.99	0.79	0.98	0.98	0.98	0.85
Pol $d = 3$	0.88	0.76	0.91	0.88	0.99	0.78	0.98	1.00	0.98	0.85
Pol $d = 4$	0.86	0.76	0.93	0.89	0.93	0.78	0.96	1.00	0.98	0.85
Pol $d = 5$	0.89	0.76	0.94	0.85	0.90	0.78	0.99	1.00	0.97	0.85
Pol $d = 6$	0.88	0.75	0.91	0.86	0.94	0.79	0.99	1.00	0.98	0.85
Pol $d = 7$	0.89	0.76	0.94	0.85	0.95	0.80	0.96	1.00	0.98	0.85
Pol $d = 8$	0.89	0.76	0.91	0.85	0.98	0.81	0.99	1.00	0.98	0.85
Pol $d = 9$	0.88	0.76	0.93	0.86	0.93	0.78	0.98	1.00	0.98	0.85
Pol $d = 10$	0.88	0.78	0.95	0.85	0.99	0.78	0.98	1.00	0.98	0.85

Table VI. **Validation set** performance of LS-SVMs on 10 binary domains, the best performances on each domain are underlined and denoted in bold face.

LS-SVM	`acr`	`bld`	`gcr`	`hea`	`ion`	`pid`	`snr`	`ttt`	`wbc`	`adu`
N_{CV}	460	230	666	180	234	512	138	638	455	33000
n	14	6	20	13	33	8	60	9	9	14
RBF	0.86	0.72	0.76	**0.83**	**0.96**	0.78	0.77	0.99	**0.97**	**0.85**
Lin	0.86	0.67	0.74	0.83	0.87	**0.78**	0.78	0.66	0.96	0.82
Pol $d = 2$	0.86	0.72	0.76	0.83	0.91	0.78	**0.82**	0.98	0.97	0.84
Pol $d = 3$	0.86	**0.73**	0.76	0.83	0.91	0.78	0.82	0.99	0.97	0.84
Pol $d = 4$	0.86	0.72	0.77	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 5$	**0.87**	0.72	0.76	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 6$	0.86	0.73	**0.77**	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 7$	0.86	0.72	0.77	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 8$	0.86	0.73	0.76	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 9$	0.86	0.73	0.77	0.83	0.78	0.78	0.81	0.99	0.97	0.84
Pol $d = 10$	0.86	0.71	0.77	0.83	0.91	0.78	0.81	**1.00**	0.97	0.84

Table VII. **Test set** performance of LS-SVMs on 10 binary domains, the best performances on each domain are underlined and denoted in bold face.

LS-SVM	`acr`	`bld`	`gcr`	`hea`	`ion`	`pid`	`snr`	`ttt`	`wbc`	`adu`
N_{test}	230	115	334	90	117	256	70	320	228	12222
n	14	6	20	13	33	8	60	9	9	14
RBF	**0.90**	0.71	**0.77**	0.87	**0.97**	0.77	0.76	0.99	**0.96**	**0.84**
Lin	0.90	**0.72**	0.77	0.86	0.88	0.77	0.74	0.67	0.96	0.82
Pol $d = 2$	0.89	0.71	0.76	0.84	0.93	**0.78**	**0.86**	0.99	0.96	0.84
Pol $d = 3$	0.90	0.71	0.77	0.84	0.93	0.77	0.83	0.99	0.96	0.85
Pol $d = 4$	0.89	0.70	0.76	0.86	0.91	0.78	0.83	**1.00**	0.96	0.84
Pol $d = 5$	0.90	0.72	0.75	0.87	0.88	0.76	0.83	1.00	0.96	0.84
Pol $d = 6$	0.89	0.71	0.76	**0.88**	0.91	0.77	0.84	**1.00**	0.96	0.85
Pol $d = 7$	0.89	0.70	0.75	0.87	0.91	0.77	0.79	**1.00**	0.96	0.85
Pol $d = 8$	0.88	0.70	0.76	0.86	0.91	0.76	0.83	**1.00**	0.96	0.85
Pol $d = 9$	0.88	0.70	0.76	0.86	0.90	0.77	0.80	0.99	0.96	0.84
Pol $d = 10$	0.89	0.71	0.75	0.88	0.94	0.78	0.80	**1.00**	0.96	0.84

Table VIII. Comparison of the 10 times randomized **test set** performance of LS-SVM and LS-SVM$_{\mathrm{F}}$ (linear, polynomial and Radial Basis Function kernel) with the performance of LDA, QDA, Logit, C4.5, oneR, IB1, IB10, NB$_{\mathrm{k}}$, NB$_{\mathrm{n}}$ and the Majority Rule classifier on 10 binary domains. The Average Accuracy (AA), Average Rank (AR) and Probability of equal medians using the Sign Test (P$_{\mathrm{ST}}$) taken over all domains are reported in the last three columns. Best performances are underlined and denoted in bold face, performances not significantly different at the 5% level are denoted in bold face, performances significantly different at the 1% level are emphasized.

	acr	bld	gcr	hea	ion	pid	snr	ttt	wbc	adu	AA	AR	P$_{\mathrm{ST}}$
N_{test}	230	115	334	90	117	256	70	320	228	12222			
n	14	6	20	13	33	8	60	9	9	14			
RBF LS-SVM	**87.0**(2.1)	**70.2**(4.1)	**76.3**(1.4)	**84.7**(4.8)	**96.0**(2.1)	**76.8**(1.7)	*73.1*(4.2)	*99.0*(0.3)	*96.4*(1.0)	*84.7*(0.3)	**84.4**	**3.5**	**0.727**
RBF LS-SVM$_{\mathrm{F}}$	**86.4**(1.9)	*65.1*(2.9)	*70.8*(2.4)	83.2(5.0)	*93.4*(2.7)	*72.9*(2.0)	*73.6*(4.6)	*97.9*(0.7)	96.8(0.7)	*77.6*(1.3)	*81.8*	8.8	**0.109**
Lin LS-SVM	**86.8**(2.2)	*65.6*(3.2)	75.4(2.3)	**84.9**(4.5)	*87.9*(2.0)	76.8(1.8)	*72.6*(3.7)	*66.8*(3.9)	*95.8*(1.0)	*81.8*(0.3)	79.4	7.7	**0.109**
Lin LS-SVM$_{\mathrm{F}}$	**86.5**(2.1)	*61.8*(3.3)	*68.6*(2.3)	*82.8*(4.4)	*85.0*(3.5)	*73.1*(1.7)	73.3(3.4)	*57.6*(1.9)	**96.9**(0.7)	*71.3*(0.3)	*75.7*	12.1	**0.109**
Pol LS-SVM	**86.5**(2.2)	**70.4**(3.7)	**76.3**(1.4)	**83.7**(3.9)	*91.0*(2.5)	**77.0**(1.8)	**76.9**(4.7)	**99.5**(0.5)	*96.4*(0.9)	*84.6*(0.3)	**84.2**	**4.1**	**0.727**
Pol LS-SVM$_{\mathrm{F}}$	**86.6**(2.2)	*65.3*(2.9)	*70.3*(2.3)	*82.4*(4.6)	*91.7*(2.6)	*73.0*(1.8)	**77.3**(2.6)	*98.1*(0.8)	**96.9**(0.7)	*77.9*(0.2)	82.0	**8.2**	**0.344**
RBF SVM	*86.3*(1.8)	**70.4**(3.2)	**75.9**(1.4)	**84.7**(4.8)	95.4(1.7)	**77.3**(2.2)	**75.0**(6.6)	*98.6*(0.5)	*96.4*(1.0)	*84.4*(0.3)	**84.4**	**4.0**	**1.000**
Lin SVM	**86.7**(2.4)	*67.7*(2.6)	75.4(1.7)	*83.2*(4.2)	*87.1*(3.4)	77.0(2.4)	*74.1*(4.2)	*66.2*(3.6)	*96.3*(1.0)	*83.9*(0.2)	*79.8*	**7.5**	0.021
LDA	*85.9*(2.2)	*65.4*(3.2)	**75.9**(2.0)	**83.9**(4.3)	*87.1*(2.3)	76.7(2.0)	*67.9*(4.9)	*68.0*(3.0)	*95.6*(1.1)	*82.2*(0.3)	*78.9*	*9.6*	*0.004*
QDA	*80.1*(1.9)	*62.2*(3.6)	*72.5*(1.4)	*78.4*(4.0)	*90.6*(2.2)	*74.2*(3.3)	*53.6*(7.4)	*75.1*(4.0)	*94.5*(0.6)	*80.7*(0.3)	*76.2*	*12.6*	*0.002*
Logit	**86.8**(2.4)	*66.3*(3.1)	**76.3**(2.1)	*82.9*(4.0)	*86.2*(3.5)	**77.2**(1.8)	*68.4*(5.2)	*68.3*(2.9)	*96.1*(1.0)	*83.7*(0.2)	79.2	7.8	**0.109**
C4.5	*85.5*(2.1)	*63.1*(3.8)	*71.4*(2.0)	*78.0*(4.2)	*90.6*(2.2)	*73.5*(3.0)	*72.1*(2.5)	*84.2*(1.6)	*94.7*(1.0)	**85.6**(0.3)	*79.9*	*10.2*	0.021
oneR	*85.4*(2.1)	*56.3*(4.4)	*66.0*(3.0)	*71.7*(3.6)	*83.6*(4.8)	*71.3*(2.7)	*62.6*(5.5)	*70.7*(1.5)	*91.8*(1.4)	*80.4*(0.3)	*74.0*	*15.5*	*0.002*
IB1	*81.1*(1.9)	*61.3*(6.2)	*69.3*(2.6)	*74.3*(4.2)	*87.2*(2.8)	*69.6*(2.4)	**77.7**(4.4)	*82.3*(3.3)	*95.3*(1.1)	*78.9*(0.2)	*77.7*	12.5	0.021
IB10	**86.4**(1.3)	*60.5*(4.4)	*72.6*(1.7)	*80.0*(4.3)	*85.9*(2.5)	*73.6*(2.4)	*69.4*(4.3)	*94.8*(2.0)	*96.4*(1.2)	*82.7*(0.3)	*80.2*	*10.4*	0.039
NB$_{\mathrm{k}}$	*81.4*(1.9)	*63.7*(4.5)	*74.7*(2.1)	83.9(4.5)	*92.1*(2.5)	75.5(1.7)	*71.6*(3.5)	*71.7*(3.1)	**97.1**(0.9)	*84.8*(0.2)	*79.7*	**7.3**	**0.109**
NB$_{\mathrm{n}}$	*76.9*(1.7)	*56.0*(6.9)	74.6(2.8)	**83.8**(4.5)	*82.8*(3.8)	75.1(2.1)	*66.6*(3.2)	*71.7*(3.1)	*95.5*(0.5)	*82.7*(0.2)	*76.6*	*12.3*	*0.002*
Maj. Rule	*56.2*(2.0)	*56.5*(3.1)	*69.7*(2.3)	*56.3*(3.8)	*64.4*(2.9)	*66.8*(2.1)	*54.4*(4.7)	*66.2*(3.6)	*66.2*(2.4)	*75.3*(0.3)	*63.2*	*17.1*	*0.002*

Table IX. Comparison of the 10 times randomized **test set** performance of LS-SVM and LS-SVM$_{\mathrm{F}}$ (linear, polynomial and Radial Basis Function kernel) with the performance of LDA, QDA, Logit, C4.5, oneR, IB1, IB10, NB$_{\mathrm{k}}$, NB$_{\mathrm{n}}$ and the Majority Rule classifier on 10 multiclass domains. The Average Accuracy (AA), Average Rank (AR) and Probability of equal medians using the Sign Test (P$_{\mathrm{ST}}$) taken over all domains are reported in the last three columns. The notation of Table VIII is used to report the results of the significance tests.

LS-SVM	bal	cmc	ims	iri	led	thy	usp	veh	wav	win	AA	AR	P$_{\mathrm{ST}}$
N_{test}	209	491	770	50	1000	2400	3298	282	1200	60			
n	4	9	18	4	7	21	256	18	19	13			
RBF LS-SVM (MOC)	*92.7*(1.0)	**54.1**(1.8)	*95.5*(0.6)	*96.6*(2.8)	*70.8*(1.4)	*96.6*(0.4)	*95.3*(0.5)	81.9(2.6)	**99.8**(0.2)	**98.7**(1.3)	**88.2**	**7.1**	**0.344**
RBF LS-SVM$_{\mathrm{F}}$ (MOC)	*86.8*(2.4)	*43.5*(2.6)	*69.6*(3.2)	**98.4**(2.1)	*36.1*(2.4)	*22.0*(4.7)	*86.5*(1.0)	*66.5*(6.1)	*99.5*(0.2)	*93.2*(3.4)	*70.2*	17.8	**0.109**
Lin LS-SVM (MOC)	*90.4*(0.8)	*46.9*(3.0)	*72.1*(1.2)	*89.6*(5.6)	*52.1*(2.2)	*93.2*(0.6)	*76.5*(0.6)	*69.4*(2.3)	*90.4*(1.1)	**97.3**(2.0)	*77.8*	*17.8*	*0.002*
Lin LS-SVM$_{\mathrm{F}}$ (MOC)	*86.6*(1.7)	*42.7*(2.0)	*69.8*(1.2)	*77.0*(3.8)	*35.1*(2.6)	*54.1*(1.3)	*58.2*(0.9)	*69.1*(2.0)	*55.7*(1.3)	*85.5*(5.1)	*63.4*	*22.4*	*0.002*
Pol LS-SVM (MOC)	*94.0*(0.8)	53.5(2.3)	*87.2*(2.6)	**96.4**(3.7)	*70.9*(1.5)	*94.7*(0.2)	*95.0*(0.8)	*81.8*(1.2)	*99.6*(0.3)	**97.8**(1.9)	87.1	**9.8**	**0.109**
Pol LS-SVM$_{\mathrm{F}}$ (MOC)	*93.2*(1.9)	*47.4*(1.6)	*86.2*(3.2)	96.0(3.7)	*67.7*(0.8)	*69.9*(2.8)	*87.2*(0.9)	*81.9*(1.3)	*96.1*(0.7)	*92.2*(3.2)	*81.8*	*15.7*	*0.002*
RBF LS-SVM (1vs1)	*94.2*(2.2)	**55.7**(2.2)	**96.5**(0.5)	**97.6**(2.3)	**74.1**(1.3)	*96.8*(0.3)	94.8(2.5)	83.6(1.3)	*99.3*(0.4)	**98.2**(1.8)	**89.1**	**5.9**	**1.000**
RBF LS-SVM$_{\mathrm{F}}$ (1vs1)	*71.4*(15.5)	*42.7*(3.7)	*46.2*(6.5)	*79.8*(10.3)	*58.9*(8.5)	*92.6*(0.2)	*30.7*(2.4)	*24.9*(2.5)	*97.3*(1.7)	*67.3*(14.6)	*61.2*	*22.3*	*0.002*
Lin LS-SVM (1vs1)	*87.8*(2.2)	*50.8*(2.4)	*93.4*(1.0)	**98.4**(1.8)	**74.5**(1.0)	*93.2*(0.3)	*95.4*(0.3)	*79.8*(2.1)	*97.6*(0.9)	**98.3**(2.5)	**86.9**	**9.7**	**0.754**
Lin LS-SVM$_{\mathrm{F}}$ (1vs1)	*87.7*(1.8)	*49.6*(1.8)	*93.4*(0.9)	**98.6**(1.3)	**74.5**(1.0)	*74.9*(0.8)	*95.3*(0.3)	*79.8*(2.2)	*98.2*(0.6)	**97.7**(1.8)	85.0	**11.1**	**0.344**
Pol LS-SVM (1vs1)	*95.4*(1.0)	*53.2*(2.2)	*95.2*(0.6)	96.8(2.3)	72.8(2.6)	88.8(14.6)	**96.0**(2.1)	82.8(1.8)	*99.0*(0.4)	**99.0**(1.4)	**87.9**	**8.9**	**0.344**
Pol LS-SVM$_{\mathrm{F}}$ (1vs1)	*56.5*(16.7)	*41.8*(1.8)	*30.1*(3.8)	*71.4*(12.4)	*32.6*(10.9)	*92.6*(0.7)	95.8(1.7)	*20.3*(6.7)	*77.5*(4.9)	*82.3*(12.2)	*60.1*	*21.9*	0.021
RBF SVM (MOC)	**99.2**(0.5)	*51.0*(1.4)	*94.9*(0.9)	96.6(3.4)	*69.9*(1.0)	*96.6*(0.2)	*95.5*(0.4)	*77.6*(1.7)	**99.7**(0.1)	**97.8**(2.1)	**87.9**	**8.6**	**0.344**
Lin SVM (MOC)	*98.3*(1.2)	*45.8*(1.6)	*74.1*(1.4)	**95.0**(10.5)	*50.9*(3.2)	*92.5*(0.3)	*81.9*(0.3)	*70.3*(2.5)	*99.2*(0.2)	97.3(2.6)	80.5	16.1	0.021
RBF SVM (1vs1)	*98.3*(1.2)	**54.7**(2.4)	*96.0*(0.4)	97.0(3.0)	*64.6*(5.6)	*98.3*(0.3)	**97.2**(0.2)	83.8(1.6)	*99.6*(0.2)	**96.8**(5.7)	**88.6**	**6.5**	**1.000**
Lin SVM (1vs1)	*91.0*(2.3)	*50.8*(1.6)	*95.2*(0.7)	**98.0**(1.9)	**74.4**(1.2)	*97.1*(0.3)	*95.1*(0.3)	*78.1*(2.4)	*99.6*(0.2)	**98.3**(3.1)	**87.8**	**7.3**	**0.754**
LDA	*86.9*(2.1)	*51.8*(2.2)	*91.2*(1.1)	**98.6**(1.0)	73.7(0.8)	*93.7*(0.3)	*91.5*(0.5)	*77.4*(2.7)	*94.6*(1.2)	**98.7**(1.5)	85.8	**11.0**	**0.109**
QDA	*90.5*(1.1)	*50.6*(2.1)	*81.8*(9.6)	**98.2**(1.8)	**73.6**(1.1)	*93.4*(0.3)	*74.7*(0.7)	**84.8**(1.5)	*60.9*(9.5)	**99.2**(1.2)	**80.8**	**11.8**	**0.344**
Logit	*88.5*(2.0)	*51.6*(2.4)	*95.4*(0.6)	**97.0**(3.9)	**73.9**(1.0)	*95.8*(0.5)	*91.5*(0.5)	*78.3*(2.3)	**99.9**(0.1)	*95.0*(3.2)	*86.7*	**9.8**	0.021
C4.5	*66.0*(3.6)	*50.9*(1.7)	96.1(0.7)	96.0(3.1)	73.6(1.3)	**99.7**(0.1)	*88.7*(0.3)	*71.1*(2.6)	**99.8**(0.1)	*87.0*(5.0)	82.9	**11.8**	**0.109**
oneR	*59.5*(3.1)	*43.2*(3.5)	*62.9*(2.4)	*95.2*(2.5)	*17.8*(0.8)	*96.3*(0.5)	*32.9*(1.1)	*52.9*(1.9)	*67.4*(1.1)	*76.2*(4.6)	*60.4*	*21.6*	*0.002*
IB1	*81.5*(2.7)	*43.3*(1.1)	**96.8**(0.6)	*95.6*(3.6)	**74.0**(1.3)	*92.2*(0.4)	*97.0*(0.2)	*70.1*(2.9)	*99.7*(0.1)	*95.2*(2.0)	**84.5**	**12.9**	**0.344**
IB10	*83.6*(2.3)	*44.3*(2.4)	*94.3*(0.7)	*97.2*(1.9)	**74.2**(1.3)	*93.7*(0.3)	*96.1*(0.3)	*67.1*(2.1)	*99.4*(0.1)	*96.2*(1.9)	84.6	**12.4**	**0.344**
NB$_{\mathrm{k}}$	*89.9*(2.0)	*51.2*(2.3)	*84.9*(1.4)	**97.0**(2.5)	**74.0**(1.2)	*96.4*(0.2)	*79.3*(0.9)	*60.0*(2.3)	*99.5*(0.1)	**97.7**(1.6)	*83.0*	12.2	0.021
NB$_{\mathrm{n}}$	*89.9*(2.0)	*48.9*(1.8)	*80.1*(1.0)	**97.2**(2.7)	**74.0**(1.2)	*95.5*(0.4)	*78.2*(0.6)	*44.9*(2.8)	*99.5*(0.1)	97.5(1.8)	*80.6*	13.6	0.021
Maj. Rule	*48.7*(2.3)	*43.2*(1.8)	*15.5*(0.6)	*38.6*(2.8)	*11.4*(0.0)	*92.5*(0.3)	*16.8*(0.4)	*27.7*(1.5)	*34.2*(0.8)	*39.7*(2.8)	*36.8*	*24.8*	*0.002*

Table X. Sparse approximation with RBF LS-SVMs by gradually pruning the support value spectrum. Pruning is stopped when the cross-validation accuracy decreases significantly at the 5% and 1% significance level (SL 5% and SL 1%), respectively. The following randomized test set performances are reported: no pruning (LS-SVM), pruning till SL 5% (LS-SVM$_{\mathrm{SL5\%}}$) and pruning till SL 1% (LS-SVM$_{\mathrm{SL1\%}}$). The corresponding number of training examples N_{CV}, $N^{(1:L)}_{\mathrm{SL5\%}}$ and $N^{(1:L)}_{\mathrm{SL1\%}}$ are also reported, together with the corresponding Percentage of Pruned Taining Examples PPTE$_{\mathrm{SL1\%}}$ and PPTE$_{\mathrm{SL5\%}}$, respectively. The notation of Table VIII is used to report the results of the significance tests.

LS-SVM	acr	bld	gcr	hea	ion	bal	cmc	ims	iri	led	AA	AR	P_{ST}
N_{test}	230	115	334	90	117	209	491	770	50	1000			
n	14	6	20	13	33	4	9	18	4	7			
LS-SVM	$\underline{\textbf{87.0}}$(2.1)	$\underline{\textbf{70.2}}$(4.1)	$\underline{\textbf{76.3}}$(1.4)	$\underline{\textbf{84.7}}$(4.8)	$\underline{\textbf{96.0}}$(2.1)	94.2(2.2)	$\underline{\textbf{55.7}}$(2.2)	$\underline{\textbf{96.5}}$(0.5)	**97.6**(2.3)	$\underline{\textbf{74.1}}$(1.3)	$\underline{\textbf{83.2}}$	$\underline{\textbf{1.3}}$	$\underline{\textbf{1.000}}$
LS-SVM$_{\mathrm{SL5\%}}$	*85.6*(2.3)	67.3(4.1)	**76.2**(1.9)	83.6(4.0)	*89.8*(1.6)	$\underline{\textbf{95.7}}$(1.2)	**54.4**(1.6)	*90.7*(1.7)	96.8(2.3)	**73.6**(2.5)	81.4	**2.2**	0.021
LS-SVM$_{\mathrm{SL1\%}}$	*85.6*(2.3)	67.3(4.1)	**75.9**(1.5)	82.8(3.6)	*89.8*(1.6)	**95.5**(1.6)	**54.4**(1.6)	*90.7*(1.7)	$\underline{\textbf{97.8}}$(2.6)	71.0(4.2)	81.1	2.5	**0.109**
$N^{(1:L)}_{CV}$	460	230	666	180	234	832	1964	9240	200	18000			
$N^{(1:L)}_{\mathrm{SL5\%}}$	149	128	344	59	56	104	1418	860	96	4360			
$N^{(1:L)}_{\mathrm{SL1\%}}$	149	128	327	57	56	60	1418	860	66	3640			
PPTE$_{\mathrm{SL5\%}}$	68%	44%	48%	67%	76%	88%	28%	91%	67%	76%			
PPTE$_{\mathrm{SL1\%}}$	68%	44%	51%	68%	76%	93%	28%	91%	52%	80%			