Model Selection for the ℓ_2-SVM by Following the Regularization Path

Rémi Bonidal
LORIA-UL
Campus Scientifique, BP 239
54506 Vandœuvre-lès-Nancy Cedex, France
(e-mail: remi.bonidal@loria.fr)

Samy Tindel
IECN-UL
Campus Scientifique, BP 70239
54506 Vandœuvre-lès-Nancy Cedex, France
(e-mail: samy.tindel@univ-lorraine.fr)

Yann Guermeur
LORIA-CNRS
Campus Scientifique, BP 239
54506 Vandœuvre-lès-Nancy Cedex, France
(e-mail: yann.guermeur@loria.fr)

December 13, 2013

Running Title: Model selection for the ℓ_2-SVM

Keywords: ℓ_2-SVM, model selection, regularization path, leave-one-out cross-validation error

Abstract

For a support vector machine, model selection consists in selecting the kernel function, the values of its parameters, and the amount of regularization. To set the value of the regularization parameter, one can minimize an appropriate objective function over the regularization path. A priori, this requires the availability of two elements: the objective function and an algorithm computing the regularization path at a reduced cost. The literature provides us with several upper bounds and estimates for the leave-one-out cross-validation error of the ℓ_2-SVM. However, no algorithm was available so far for fitting the entire regularization path of this machine. In this article, we introduce the first algorithm of this kind. It is involved in the specification of new methods to tune the corresponding penalization coefficient, whose objective function is a leave-one-out error bound or estimate. From a computational point of view, these methods appear especially appropriate when the Gram matrix is of low rank. A comparative study involving state-of-the-art alternatives provides us with an empirical confirmation of this advantage.

1 Introduction

During the last decade, Vapnik's main model of soft margin pattern recognition support vector machine (SVM) [12], hereafter referred to as the ℓ_1-SVM, has become one of the most popular methods to compute dichotomies. Several variants exist, such as the ℓ_2-SVM [12] and the least squares SVM (LS-SVM) [30], which have also been the subject of extensive studies. Two main reasons can be put forward to explain this success. On the first hand, these machines perform well in practice. On the other hand, their use is, at least in principle, very simple. These advantages are tempered by the fact that in spite of important efforts performed over the years [9, 19, 4, 24], model selection for SVMs remains an open problem. Generally speaking, model selection raises two main issues. The first one regards the criterion used to evaluate the quality of a model. The second one is the search for the model optimizing this criterion.

During the last decade, a great many methods have been proposed to estimate the generalization performance of the SVMs. As usual, the solution of reference is the V-fold cross-validation [28, 5], with its extremal variant, the leave-one-out procedure, providing

an almost unbiased estimator of the generalization error [25]. Since computing the leave-one-out cross-validation error can be practically intractable, one usually resorts to upper bounds or estimates. This is known to provide good results in practice (see for instance [9]). Among the bounds, the radius-margin one [31] provides a good compromise between efficiency and computational complexity [9, 10]. It applies to the hard margin machine, and, by extension, to the ℓ_2-SVM. The span bound [9] is tighter, but at the expense of a higher running time complexity.

In the same way as a great many criteria are available for model selection, the options to optimize them can be multiple. A solution that is always available is the most naive (and practically the most expensive) one: a grid search over the parameter space. Since some criteria are differentiable, such as the radius-margin bound and the leave-one-out test error prediction using differentiable spans (named *span prediction with regularization* in [9]), methods based on a gradient descent have also been developed [9, 10]. Usually, these local methods provide satisfactory results at a low cost, but they can also get stuck in plateaus of the objective function, or converge to suboptimal minima (when the objective function is not convex). A good illustration of this phenomenon is provided in [10]. In [22], the first algorithm for computing all the solutions of an ℓ_1-SVM along its regularization path was proposed. Experimental results show that its computational complexity is only slightly superior to that of a single training of the corresponding machine. Furthermore, its use makes it possible to find a global minimum of the selected criterion.

Given the positive judgment passed by the literature on the radius-margin bound, it appears interesting to derive an algorithm fitting the entire regularization path of the ℓ_2-SVM, and use it to perform a comparative study of different criteria (radius-margin bound, test error predictions) available to tune the value of the corresponding penalization coefficient. This is the subject of this paper. This algorithm is based on a continuation technique [1] that makes use of an active set method [16].

The organization of the paper is as follows. Section 2 introduces our algorithm performing the exploration of the regularization path for the ℓ_2-SVM. Section 3 details upper bounds on the generalization error of this machine, as well as estimates of this quantity. It addresses the integration of their computation in the framework of the regularization path algorithm. Section 4 presents experimental results on both synthetic and real data sets. The path following algorithm is first assessed alone, in terms of fitness and speed. It is then reassessed for model selection, in the framework of a comparative study with a gradient based method. Lastly, we draw conclusions and outline our ongoing research in

Section 5.

2 Regularization Path for the ℓ_2-SVM

We are interested in binary discrimination. Each object is represented by its description $x \in \mathcal{X}$ and the set $\mathcal{Y}$ of the categories y can be identified with the set $\{-1, 1\}$. The assignment of the descriptions to the categories is performed by means of a *classifier*, i.e., a real-valued function on $\mathcal{X}$. For such a function g, the corresponding *decision rule* f is defined as follows:

$$\forall x \in \mathcal{X}, \quad \begin{cases} g(x) < 0 \Longleftrightarrow f(x) = -1 \\ g(x) = 0 \Longleftrightarrow f(x) = * \\ g(x) > 0 \Longleftrightarrow f(x) = 1 \end{cases} ,$$

where $*$ denotes a dummy category introduced to deal with the cases of ex æquo. Thus, the example $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is correctly classified by g if and only if $yg(x) > 0$. In the sequel, the family of classifiers of interest is the class $\mathcal{H}$ of the functions implemented by an SVM. Let κ be a real-valued positive type function [3] on $\mathcal{X}^2$, also named kernel, and let $(\mathbf{H}_\kappa, \langle \cdot, \cdot \rangle_\kappa)$ be the corresponding reproducing kernel Hilbert space (RKHS) [2, 3]. Let $\{1\}$ be the one-dimensional space of real-valued constant functions on $\mathcal{X}$. Then, the formula giving $\mathcal{H}$ is

$$\mathcal{H} = (\mathbf{H}_\kappa, \langle \cdot, \cdot \rangle_\kappa) + \{1\} \quad .$$

The reproducing property allows us to write the functions of $\mathcal{H}$ as affine functions on $\mathbf{H}_\kappa$, i.e.,

$$\forall h \in \mathcal{H},\ \forall x \in \mathcal{X},\ \ h(x) = \bar{h}(x) + b = \langle \bar{h}, \kappa_x \rangle_\kappa + b,$$

where $\bar{h} \in \mathbf{H}_\kappa$, $\kappa_x = \kappa(x, \cdot) \in \mathbf{H}_\kappa$, and $b \in \mathbb{R}$. $\mathbf{H}_\kappa$ is thus one of the possible feature spaces, associated with the feature map given by

$$\begin{array}{rccc} \Phi: & \mathcal{X} & \to & \mathbf{H}_\kappa \\ & x & \mapsto & \kappa_x \end{array} .$$

To keep this article self-contained, the ℓ_2-SVM is now briefly introduced. Then, following the structure of [22], the three constitutive elements of our regularization path algorithm are detailed:

- computation of the Lagrange multipliers,

- identification of the breakpoints (transitions between regimes),
- initialization.

The main difference in the way they combine is that the identification of the breakpoints makes use, in an iterative manner, of the computation of the Lagrange multipliers. Thus these elements combine themselves in a continuation method similar to the algorithm introduced in [27]. The section concludes with the flowchart of the algorithm and an analysis of its implementation.

2.1 Learning Problem of the ℓ_2-SVM

The ℓ_2-SVM is the variant of the standard bi-class SVM obtained by replacing in the objective function of the primal formulation of the learning problem the ℓ_1 norm of the vector of slack variables $\xi = (\xi_i)_{1 \leqslant i \leqslant m}$ with the square of the ℓ_2 norm of the same vector. A direct consequence is the fact that it is no longer necessary to explicitly consider the constraints of nonnegativity of the slack variables. Thus, given a training set $d_m = \{(x_i, y_i) : 1 \leqslant i \leqslant m\}$, the primal formulation of the learning problem corresponds to the following convex quadratic programming (QP) problem.

Problem 1 (Learning problem of the ℓ_2-SVM, primal formulation)

$$\min_{h,\xi} \left\{ \|\xi\|_2^2 + \frac{\lambda}{2} \left\|\bar{h}\right\|_\kappa^2 \right\}$$

$$s.t.\ \forall i \in [\![1, m]\!],\ \ y_i h(x_i) \geqslant 1 - \xi_i\ \ .$$

Let $y = (y_i)_{1 \leqslant i \leqslant m} \in \{-1, 1\}^m$ be the vector of the labels of the training examples. For n in $\mathbb{N}^*$, let $\mathcal{M}_{n,n}(\mathbb{R})$ be the algebra of $n \times n$ matrices over $\mathbb{R}$. Let $H = (h_{i,j})_{1 \leqslant i,j \leqslant m} \in \mathcal{M}_{m,m}(\mathbb{R})$ be the Hessian matrix with $h_{i,j} = y_i y_j \kappa(x_i, x_j)$. For $\lambda \in \mathbb{R}_+^*$, let $H(\lambda) = (h_{i,j}(\lambda))_{1 \leqslant i,j \leqslant m} \in \mathcal{M}_{m,m}(\mathbb{R})$ be the matrix deduced from H by replacing the kernel κ with the function κ_λ given by:

$$\forall (i, j) \in [\![1, m]\!]^2,\ \ \kappa_\lambda(x_i, x_j) = \kappa(x_i, x_j) + \frac{\lambda}{2}\delta_{i,j},$$

where δ is the Kronecker symbol. Since κ_λ is not defined on $\mathcal{X}^2$, it does not satisfy the definition of a kernel. Nevertheless, by a slight misuse of language, it will be called kernel by analogy, due to its role in the hard margin SVM training problem. Let $\beta = (\beta_i)_{1 \leqslant i \leqslant m} \in \mathbb{R}_+^m$ be the vector of the Lagrange multipliers associated with the constraints

of good classification and $\alpha = \frac{1}{\lambda}\beta$. Then, the dual of Problem 1 is the following convexe QP problem.

Problem 2 (Learning problem of the ℓ_2-SVM, dual formulation)

$$\max_{\alpha} \left\{ -\frac{1}{2}\alpha^T H(\lambda)\alpha + 1_m^T \alpha \right\}$$

$$s.t. \quad \begin{cases} \forall i \in [\![1, m]\!], \;\; \alpha_i \geqslant 0 \\ y^T \alpha = 0 \end{cases} .$$

Problem 2 is also the dual formulation of the learning problem of a hard margin SVM "using κ_λ as kernel". This property is important since it implies that results holding for the hard margin machine also hold for the ℓ_2-SVM. As pointed out in the introduction, among these results is the radius-margin bound. For a given value of λ, let $(h_\lambda, \xi(\lambda)) = \left(\bar{h}_\lambda, b_\lambda, \xi(\lambda)\right)$ be the optimal solution of Problem 1 and $\alpha(\lambda) = (\alpha_i(\lambda))_{1 \leqslant i \leqslant m}$ the optimal solution of Problem 2. Noticeable equations obtained when deriving Problem 2 (applying the Lagrangian duality) are:

$$\xi(\lambda) = \frac{\lambda}{2}\alpha(\lambda) \tag{1}$$

and

$$h_\lambda = \sum_{i=1}^{m} \alpha_i(\lambda)\, y_i \kappa_{x_i} + \alpha_0(\lambda), \tag{2}$$

with $\alpha_0(\lambda) = b_\lambda$. A direct consequence of (2) and the definition of the matrix H is that

$$\forall i \in [\![1, m]\!], \;\; y_i h_\lambda(x_i) = H_{i,.}\alpha(\lambda) + y_i \alpha_0(\lambda) \tag{3}$$

where $H_{i,.}$ stands for the row of index i of H ($H_{.,j}$ would stand for its column of index j). Let $\tilde{h}_\lambda$ be the function computed by the hard margin machine associated with Problem 2. Then,

$$\begin{aligned} \forall i \in [\![1, m]\!], \;\; \tilde{h}_\lambda(x_i) &= \sum_{j=1}^{m} \left\{ \alpha_j(\lambda)\, y_j \kappa(x_j, x_i) + \frac{\lambda}{2}\alpha_j(\lambda)\, y_j \delta_{i,j} \right\} + \alpha_0(\lambda) \\ &= h_\lambda(x_i) + \frac{\lambda}{2}\alpha_i(\lambda)\, y_i = h_\lambda(x_i) + y_i \xi_i(\lambda) \;\; . \end{aligned}$$

2.2 Partitioning the Training Set

To follow the path of interest, taking our inspiration from the algorithm introduced in [22], we focus on the study of the way the values of the positive Lagrange multipliers vary as a function of λ, while the set of null multipliers remains unchanged. This requires us to identify both sets. The Kuhn-Tucker (KT) optimality conditions provide us with such information. The KT complementary conditions corresponding to the two machines are given by:

$$\forall i \in [\![1, m]\!], \quad \alpha_i(\lambda) \left[y_i h_\lambda(x_i) - 1 + \xi_i(\lambda) \right] = \alpha_i(\lambda) \left[y_i \tilde{h}_\lambda(x_i) - 1 \right] = 0 \ . \tag{4}$$

It springs from (1) and (4) that for each example, there are three possibilities:

- $y_i h_\lambda(x_i) < 1$: the example is inside the margin or misclassified, so $\xi_i(\lambda) > 0$ and $\alpha_i(\lambda) > 0$,
- $y_i h_\lambda(x_i) = 1$: the example is on the margin and $\alpha_i(\lambda) = 0$,
- $y_i h_\lambda(x_i) > 1$: the example is correctly classified and outside the margin, $\alpha_i(\lambda) = 0$.

To sum up, $\alpha_i(\lambda) > 0$ if and only if $y_i h_\lambda(x_i) < 1$. This leads to the following partition of $[\![1, m]\!]$:

$$\begin{aligned} \mathcal{E}(\lambda) &= \{ i \in [\![1, m]\!] : y_i h_\lambda(x_i) < 1 \} \\ \mathcal{I}(\lambda) &= \{ i \in [\![1, m]\!] : y_i h_\lambda(x_i) \geqslant 1 \} \ . \end{aligned}$$

This partition is reminiscent of active set methods (see for instance Section 10.3 of [16]), since $i \in \mathcal{I}(\lambda)$ if and only if the constraint $\alpha_i \geqslant 0$ is active at the optimum of Problem 2. More precisely, solving Problem 2 under the assumption that the partition associated with the optimum is known amounts to performing the last step of an active set mehod. It is well known that for a convex QP problem, each of these steps corresponds to solving a linear system. We now exhibit the linear system of interest, which calls for the introduction of additional notation. For ease of notation, in the sequel, we use $\mathcal{E}$ (respectively $\mathcal{I}$) in place of $\mathcal{E}(\lambda)$ (respectively $\mathcal{I}(\lambda)$) when no confusion is possible. The cardinalities of these sets are respectively denoted by $m_\mathcal{E}$ and $m_\mathcal{I}$. The examples are reordered in such a way that the first of them are those belonging to $\mathcal{E}$. This enables us to introduce compact notations with obvious meaning:

$$y = (y_i)_{1 \leqslant i \leqslant m} = \begin{pmatrix} y_\mathcal{E}^T & y_\mathcal{I}^T \end{pmatrix}^T ,$$

$$\alpha(\lambda) = (\alpha_i(\lambda))_{1 \leqslant i \leqslant m} = \begin{pmatrix} \alpha_{\mathcal{E}}(\lambda)^T & \alpha_{\mathcal{I}}(\lambda)^T \end{pmatrix}^T,$$

$$d_{\mathcal{E}(\lambda)} = \{(x_i, y_i) \ : \ i \in \mathcal{E}(\lambda)\},$$

and

$$H = \begin{pmatrix} H_{\mathcal{E},\mathcal{E}} & H_{\mathcal{E},\mathcal{I}} \\ H_{\mathcal{I},\mathcal{E}} & H_{\mathcal{I},\mathcal{I}} \end{pmatrix},$$

where the submatrix $H_{\mathcal{K},\mathcal{L}}$ with $(\mathcal{K},\mathcal{L}) \in \{\mathcal{E},\mathcal{I}\}^2$ is the matrix $(h_{i,j})_{i\in\mathcal{K},j\in\mathcal{L}} = (y_i y_j \kappa(x_i, x_j))_{i\in\mathcal{K},j\in\mathcal{L}}$. The matrix $H(\lambda)$ is split in the same way as H.

2.3 Analytical Expression of the Lagrange Multipliers

To derive an analytical expression of the Lagrange multipliers, we make use of the following implication, which is a consequence of (4):

$$i \in \mathcal{E} \Longrightarrow y_i \tilde{h}_\lambda(x_i) = 1 \ .$$

In other words, we work with the hard margin machine. According to our notation, the equality constraint of Problem 2 and the KT conditions (4) associated with the examples in $d_{\mathcal{E}}$ become:

$$\begin{cases} y_{\mathcal{E}}^T \alpha_{\mathcal{E}}(\lambda) + y_{\mathcal{I}}^T \alpha_{\mathcal{I}}(\lambda) = 0 \\ H_{\mathcal{E},\mathcal{E}}(\lambda)\,\alpha_{\mathcal{E}}(\lambda) + H_{\mathcal{E},\mathcal{I}}(\lambda)\,\alpha_{\mathcal{I}}(\lambda) + \alpha_0(\lambda)\, y_{\mathcal{E}} = 1_{m_{\mathcal{E}}} \end{cases}.$$

Due to the fact that $\alpha_{\mathcal{I}}(\lambda) = 0$, these equations simplify into

$$\begin{cases} y_{\mathcal{E}}^T \alpha_{\mathcal{E}}(\lambda) = 0 \\ H_{\mathcal{E},\mathcal{E}}(\lambda)\,\alpha_{\mathcal{E}}(\lambda) + \alpha_0(\lambda)\, y_{\mathcal{E}} = 1_{m_{\mathcal{E}}} \end{cases}. \tag{5}$$

By noting

$$A_{\mathcal{E}}(\lambda) = \begin{pmatrix} 0 & y_{\mathcal{E}}^T \\ y_{\mathcal{E}} & H_{\mathcal{E},\mathcal{E}}(\lambda) \end{pmatrix} = \begin{pmatrix} 0 & y_{\mathcal{E}}^T \\ y_{\mathcal{E}} & H_{\mathcal{E},\mathcal{E}} + \frac{\lambda}{2} I_{m_{\mathcal{E}}} \end{pmatrix},$$

$$C_{\mathcal{E}} = \begin{pmatrix} 0 & 1_{m_{\mathcal{E}}}^T \end{pmatrix}^T,$$

and

$$\alpha_{\mathcal{E}}^a(\lambda) = \begin{pmatrix} \alpha_0(\lambda) & \alpha_{\mathcal{E}}(\lambda)^T \end{pmatrix}^T,$$

we obtain the following proposition.

Proposition 1 *For all $\lambda \in \mathbb{R}_+^*$, the vector $\alpha_{\mathcal{E}}^a(\lambda)$ is a solution of the linear system:*

$$A_{\mathcal{E}}(\lambda)\,\alpha_{\mathcal{E}}^a(\lambda) = C_{\mathcal{E}} \ . \tag{6}$$

This formulation also appears equivalent to the one derived in Appendix 2.B. of [8]. Once $\mathcal{E}$ is known, training the ℓ_2-SVM boils down to solving (6). The main difference with the corresponding formula obtained for the ℓ_1-SVM (see for instance Section 4 in [22]) rests in the dependency of the matrix of the linear system on λ. This implies that the Lagrange multipliers do not vary linearly as a function of λ anymore. We now discuss the connection of the linear system of Proposition 1 with the learning problem of an LS-SVM. The interest of this discussion is twofold. First, it will be at the basis of the method proposed to compute the Lagrange multipliers. Second, it will highlight a link between the leave-one-out cross-validation error of the LS-SVM and the *leave-one-out test error prediction* of the ℓ_2-SVM defined in [32]. The (primal) objective function of the LS-SVM is the same as the one of Problem 1. The difference between the two learning problems rests in the fact that for the LS-SVM, the constraints of correct classification are equality constraints (with the consequence that the slack variables are not constrained in sign).

Problem 3 (Learning problem of the LS-SVM, primal formulation)

$$\min_{h,\xi} \left\{ \|\xi\|_2^2 + \frac{\lambda}{2} \left\|\bar{h}\right\|_\kappa^2 \right\}$$

$$s.t. \quad \forall i \in [\![1, m]\!], \ \ y_i h(x_i) = 1 - \xi_i \ .$$

Suykens and Vandewalle have shown that solving Problem 3 is equivalent to solving the following linear system (Equation 20 in [30]):

$$\begin{pmatrix} 0 & y^T \\ y & H + \frac{\lambda}{2} I_m \end{pmatrix} \begin{pmatrix} \alpha_0(\lambda) \\ \alpha(\lambda) \end{pmatrix} = \begin{pmatrix} 0 \\ 1_m \end{pmatrix} \ .$$

Thus, solving (6) can alternatively be seen as training an LS-SVM on $d_{\mathcal{E}(\lambda)}$. This allows us to state the following proposition.

Proposition 2 *Training an ℓ_2-SVM on d_m reduces itself to training an LS-SVM on $d_{\mathcal{E}(\lambda)}$ once the set $\mathcal{E}(\lambda)$ is known.*

We now address the practical resolution of the linear system.

2.4 Practical Computation of the Lagrange Multipliers

Our path-following strategy for setting the value of the regularization coefficient involves multiple solutions of the linear system (6), for different values of λ. This calls for a dedicated algorithm. In this section, we first reformulate (6) using a standard technique, and then make use of this reformulation to derive an algorithm solving it at a reduced computational cost in the regularization path framework.

2.4.1 Stand-Alone Solution for a Given Value of λ

We take our inspiration from the approach of [29], which uses a classical trick in active set method algorithms. We first note that since $H_{\mathcal{E},\mathcal{E}}$ is symmetric positive semi-definite, $H_{\mathcal{E},\mathcal{E}}(\lambda)$ is symmetric positive definite (SPD) and thus invertible. We want to reformulate (6) so as to make $H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1}$ appear. It springs from the second equation of (5) that:

$$\alpha_{\mathcal{E}}(\lambda) = H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1}(1_{m_{\mathcal{E}}} - \alpha_0(\lambda)\, y_{\mathcal{E}}) \quad . \tag{7}$$

By plugging (7) in the first equation of (5), we have:

$$y_{\mathcal{E}}^T H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} 1_{m_{\mathcal{E}}} = \alpha_0(\lambda)\, y_{\mathcal{E}}^T H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} y_{\mathcal{E}} \quad . \tag{8}$$

Still following [29], in order to get compact expressions for $\alpha_0(\lambda)$ and $\alpha_{\mathcal{E}}(\lambda)$, we introduce two vectors of $\mathbb{R}^{m_{\mathcal{E}}}$, ν and ρ, such that

$$\begin{cases} H_{\mathcal{E},\mathcal{E}}(\lambda)\,\nu = y_{\mathcal{E}} \\ H_{\mathcal{E},\mathcal{E}}(\lambda)\,\rho = 1_{m_{\mathcal{E}}} \end{cases} . \tag{9}$$

By substitution into (7) and (8) we get

$$\begin{cases} \alpha_{\mathcal{E}}(\lambda) = \rho - \alpha_0(\lambda)\,\nu \\ y_{\mathcal{E}}^T \rho = \alpha_0(\lambda)\, y_{\mathcal{E}}^T \nu \end{cases} .$$

Thus, the expressions of $\alpha_0(\lambda)$ and $\alpha_{\mathcal{E}}(\lambda)$ as a function of ν and ρ are:

$$\begin{cases} \alpha_0(\lambda) = \frac{y_{\mathcal{E}}^T \rho}{y_{\mathcal{E}}^T \nu} \\ \alpha_{\mathcal{E}}(\lambda) = \rho - \alpha_0(\lambda)\,\nu \end{cases} . \tag{10}$$

All in all, solving (6) for a given value of λ boils down to computing the corresponding values of ν and ρ. Since matrix $H_{\mathcal{E},\mathcal{E}}(\lambda)$ is SPD, this can be done by applying a Cholesky decomposition, such as the Gaxpy Cholesky one (see for instance [20]), whose complexity is $\frac{1}{3}m_{\mathcal{E}}^3$. However, significant gains in time can result from the global handling of the sequence of systems (9) (parameterized by λ) that are actually to be solved all along the path. We now describe a reduced-cost algorithm of this kind.

2.4.2 Computation of a Series of Solutions

Our method makes central use of a proposition that states that if the matrix H can be approximated by a matrix of low rank (independent of m), then the complexity of solving System (9) grows linearly with $m_{\mathcal{E}}$. Thus, we first introduce a low-rank approximation of H and then use it for calculating the Lagrange multipliers. In fact, the algorithm will actually follow the regularization path of the kernel associated with this approximation. In order to avoid numerical difficulties, we introduce a small positive constant: ε. It is supposed to be large enough so that $H(\varepsilon)$ is "numerically SPD" (with "not too small" eigenvalues). To derive an approximate solution of (9) for $\lambda_0 > \varepsilon$ such that $\mathcal{E}(\lambda_0)$ is known, we use a low-rank approximation of $H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0)}(\varepsilon)$, approximation which can be directly deduced from a low-rank approximation of $H(\varepsilon)$. The approximation of $H(\varepsilon)$ (and thus H) we consider is a matrix $H_r(\varepsilon)$ satisfying

$$H_r(\varepsilon) = R_r R_r^T$$

with $R_r \in \mathcal{M}_{m,r}(\mathbb{R})$. Note that this factorization can actually rest on the fact that $H(\varepsilon)$ is SPD. Let us reorder the training examples and decompose the corresponding matrix R_r according to the principle introduced in Section 2.2. Thus, $R_r = \begin{pmatrix} R_{\mathcal{E}(\lambda_0)} \\ R_{\mathcal{I}(\lambda_0)} \end{pmatrix}$, with $R_{\mathcal{E}(\lambda_0)} \in \mathcal{M}_{m_{\mathcal{E}(\lambda_0)},r}(\mathbb{R})$ and $R_{\mathcal{I}(\lambda_0)} \in \mathcal{M}_{m_{\mathcal{I}(\lambda_0)},r}(\mathbb{R})$, and we get

$$H_r(\varepsilon) = \begin{pmatrix} H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\varepsilon) & H_{\mathcal{E}(\lambda_0),\mathcal{I}(\lambda_0),r}(\varepsilon) \\ H_{\mathcal{I}(\lambda_0),\mathcal{E}(\lambda_0),r}(\varepsilon) & H_{\mathcal{I}(\lambda_0),\mathcal{I}(\lambda_0),r}(\varepsilon) \end{pmatrix} = \begin{pmatrix} R_{\mathcal{E}(\lambda_0)} R_{\mathcal{E}(\lambda_0)}^T & R_{\mathcal{E}(\lambda_0)} R_{\mathcal{I}(\lambda_0)}^T \\ R_{\mathcal{I}(\lambda_0)} R_{\mathcal{E}(\lambda_0)}^T & R_{\mathcal{I}(\lambda_0)} R_{\mathcal{I}(\lambda_0)}^T \end{pmatrix} .$$

By construction,

$$H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0)}(\lambda_0) = H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0)}(\varepsilon) + \frac{\lambda_0 - \varepsilon}{2} I_{m_{\mathcal{E}(\lambda_0)}} ,$$

thus, $H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0)}(\lambda_0)$ can be approximated by

$$H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\lambda_0,\varepsilon) = H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\varepsilon) + \frac{\lambda_0-\varepsilon}{2} I_{m_{\mathcal{E}(\lambda_0)}} \ .$$

The point of that approximation is that in the end, we work with a matrix, $R_{\mathcal{E}(\lambda_0)}$, which can be directly derived from the initial matrix R_r, i.e., from one single low-rank approximation of $H(\varepsilon)$. The fact that $H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\varepsilon)$ is not the matrix we would have obtained by approximating directly $H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0)}(\varepsilon)$ raises no difficulty. Indeed, we only make use of the factorization of the matrix and not of the nature of the approximation. Furthermore, this matrix is regular.

Proposition 3 *Let us consider the system of linear equations*

$$H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\lambda_0,\varepsilon)\, v = z,$$

with $H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\lambda_0,\varepsilon) \in \mathcal{M}_{m_{\mathcal{E}(\lambda_0)},m_{\mathcal{E}(\lambda_0)}}(\mathbb{R})$ *being the sum of a rank* r *matrix and a scaled identity matrix. The computational complexity of solving this system is linear in* $m_{\mathcal{E}(\lambda_0)}$*, precisely* $O(m_{\mathcal{E}(\lambda_0)}r^2 + r^3)$*.*

Proof

By applying the Sherman-Morrison-Woodbury identity (see for instance [20]), we obtain:

$$\begin{aligned}
\left(H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\varepsilon) + \frac{\lambda_0-\varepsilon}{2} I_{m_{\mathcal{E}(\lambda_0)}}\right)^{-1} &= \left(R_{\mathcal{E}(\lambda_0)} R_{\mathcal{E}(\lambda_0)}^T + \frac{\lambda_0-\varepsilon}{2} I_{m_{\mathcal{E}(\lambda_0)}}\right)^{-1} \\
&= \frac{2}{\lambda_0-\varepsilon} I_{m_{\mathcal{E}(\lambda_0)}} - \frac{4}{(\lambda_0-\varepsilon)^2} R_{\mathcal{E}(\lambda_0)} \left(I_r + \frac{2}{\lambda_0-\varepsilon} R_{\mathcal{E}(\lambda_0)}^T R_{\mathcal{E}(\lambda_0)}\right)^{-1} R_{\mathcal{E}(\lambda_0)}^T \\
&= \frac{2}{\lambda_0-\varepsilon} I_{m_{\mathcal{E}}(\lambda_0)} - \frac{4}{(\lambda_0-\varepsilon)^2} R_{\mathcal{E}(\lambda_0)} T(\lambda_0)^{-1} R_{\mathcal{E}(\lambda_0)}^T \qquad (11)
\end{aligned}$$

with

$$T(\lambda_0) = I_r + \frac{2}{\lambda_0-\varepsilon} R_{\mathcal{E}(\lambda_0)}^T R_{\mathcal{E}(\lambda_0)} \ .$$

Solving the system of interest boils down to solving a system involving the matrix $T(\lambda_0)$, of size r, and performing a series of matrix-vector calculations from the right to the left. Once $T(\lambda_0)$ is obtained in $O\left(m_{\mathcal{E}(\lambda_0)}r^2\right)$, the system of size r is solved by a Cholesky decomposition in only $\frac{1}{3}r^3$ operations (see the preceding section). Since the matrix multiplications of (11) involve matrices of size lower than $\left(m_{\mathcal{E}(\lambda_0)}, r\right)$, their complexity does not exceed $O(m_{\mathcal{E}(\lambda_0)}r^2)$. Summing the two complexities provides the overall complexity in

$O(m_{\mathcal{E}(\lambda_0)}r^2 + r^3)$. This concludes the proof. ■

To derive an approximate solution of (9) for $\lambda = \lambda_0$, the matrix $H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0)}(\lambda_0)$ is replaced with $H_{\mathcal{E}(\lambda_0),\mathcal{E}(\lambda_0),r}(\varepsilon) + \frac{\lambda_0-\varepsilon}{2} I_{m_{\mathcal{E}(\lambda_0)}}$. The corresponding values of ν and ρ are thus obtained by solving a system involving $T(\lambda_0)$. In order to obtain triangular systems for both equations, a Cholesky decomposition of this matrix is performed so that $T(\lambda_0) = L^T_{\mathcal{E}(\lambda_0)} L_{\mathcal{E}(\lambda_0)}$. Thanks to Proposition 3, the complexity of computing ν and ρ reduces from $O(m^3_{\mathcal{E}})$ to $O(m_{\mathcal{E}}r^2 + r^3)$, hence the linear complexity in $m_{\mathcal{E}}$ announced at the beginning of the section. This establishes the advantage of our algorithm computing globally approximate solutions for all the systems (9) of interest compared to solving independently these systems. By abuse of notation, we keep using $\alpha^a_{\mathcal{E}}(\lambda_0) = \begin{pmatrix} \alpha_0(\lambda_0) & \alpha_{\mathcal{E}}(\lambda_0)^T \end{pmatrix}^T$ to designate the resulting approximate solution of (6) (for $\lambda = \lambda_0$). Details regarding complexity and the consequences of the approximation are given in Section 2.7. This approach assumes that $\mathcal{E}(\lambda_0)$ is known. The following section presents our method for updating the set $\mathcal{E}$.

2.5 Detecting Changes in the Partition of the Training Set

Let $\left(\lambda^l\right)_{l\in\mathbb{N}^*}$ be the strictly decreasing sequence of values of the regularization coefficient associated with changes in the sets $\mathcal{E}$ and $\mathcal{I}$. These events can be directly inferred from the definitions of the sets. Indeed, if the sequence is known up to its term of index l, finding λ^{l+1} boils down to identifying the largest value λ in the interval $\left(0, \lambda^l\right)$ for which there is a new index $i \in [\![1, m]\!]$ such that $y_i h_\lambda(x_i) = 1$. Given (3), this corresponds to selecting

$$\lambda^{l+1} = \max_{\left\{i:\lambda_i^{l+1}<\lambda^l\right\}} \lambda_i^{l+1}$$

with λ_i^{l+1} being the root of

$$H_{i,\mathcal{E}(\lambda^l)}\alpha_{\mathcal{E}(\lambda^l)}(\lambda) + y_i\alpha_0(\lambda) = 1 \tag{12}$$

for all $i \in [\![1, m]\!]$.

Unfortunately this approach, requiring to solve m nonlinear equations at each step, is numerically intractable for large values of $m_{\mathcal{E}}$, and we will see in Section 2.6 that the extreme case $\mathcal{E}(\lambda) = [\![1, m]\!]$ is met in practice. In order to avoid solving nonlinear equations, we resort to a linear approximation of h_λ thanks to a first order Taylor expansion. Figure 1

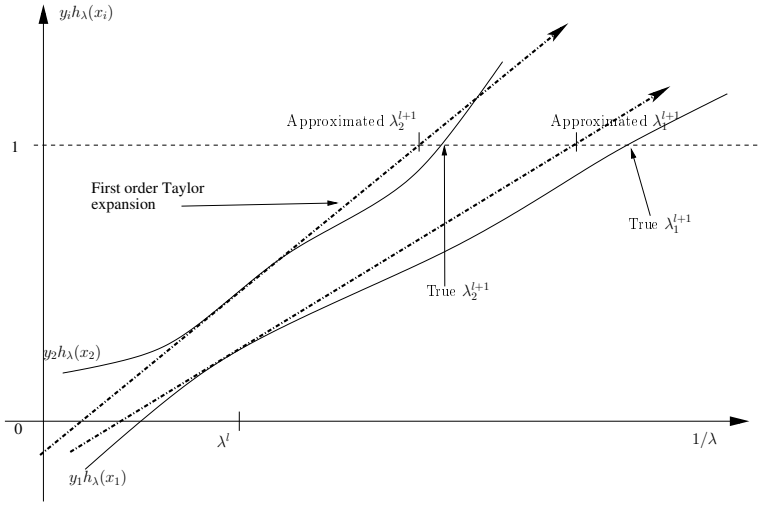

Figure 1: Illustration of the first order Taylor expansion of $y_i h_\lambda(x_i)$ for two different examples.

presents the result of this expansion. The value of λ_i^{l+1} given by the approximation is:

$$\forall i \in [\![1, m]\!], \quad \lambda_i^{l+1} = \lambda^l - \frac{1 - y_i h_{\lambda^l}(x_i)}{\left.\frac{\partial(1 - y_i h_\lambda(x_i))}{\partial \lambda}\right|_{\lambda=\lambda^l}} = \lambda^l + \frac{1 - y_i h_{\lambda^l}(x_i)}{y_i \left.\frac{\partial h_\lambda(x_i)}{\partial \lambda}\right|_{\lambda=\lambda^l}} \quad .$$

Thanks to (3), the expression of $\frac{\partial h_\lambda(x_i)}{\partial \lambda}$ is easily obtained once $\frac{\partial \alpha_0(\lambda)}{\partial \lambda}$ and $\frac{\partial \alpha_\mathcal{E}(\lambda)}{\partial \lambda}$ are known. Taking the derivative of h_λ with respect to λ is correct since the QP that has $H_r(\lambda)$ as Hessian matrix is convex. After some algebra, these quantities are derived from (9):

$$\begin{cases} \frac{\partial \alpha_0(\lambda)}{\partial \lambda} = -\frac{1}{2} \frac{\nu^T \rho y_\mathcal{E}^T \nu - \nu^T \nu y_\mathcal{E}^T \rho}{\left(y_\mathcal{E}^T \nu\right)^2} \\ \frac{\partial \alpha_\mathcal{E}(\lambda)}{\partial \lambda} = -\frac{1}{2} H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} \alpha_\mathcal{E}(\lambda) - \frac{\partial \alpha_0(\lambda)}{\partial \lambda} \nu \end{cases} \quad . \tag{13}$$

However, the use of the approximation of h_λ calls for another definition of λ^{l+1}. Indeed, it is possible for the algorithm to perform an *empty* step (no changes during the step) or a *too big* step (too many changes during the step). We experimentally found the following rule of thumb to be especially efficient: λ^{l+1} is chosen so that 2% of the indices of $\mathcal{E}(\lambda^l)$ satisfy $\lambda^{l+1} < \lambda_i^{l+1} < \lambda^l$. By doing so, we manage to do small non-empty steps. Futhermore, due to the possible flatness of h_λ, the approximation may still provide an inappropriate step size. In order to overcome this problem, we limit the largest possible step by chosing λ^{l+1} at least equal to $\frac{1}{2}\lambda^l$. Our experiments have shown that such a flatness problem only occurs when λ is very large. The prediction of the partition at λ^{l+1} is based on the sign of the first order Taylor expansion of $1 - y_i h_\lambda(x_i)$. A priori, due to the nonlinear behavior of the

Lagrange multipliers, this prediction must be corrected. The exact partition is obtained thanks to the approach implemented by the active set methods. For a given partition, the values of the Lagrange multipliers are computed, and the consistency of the result is checked. In case of inconsistency, a new partition is inferred from the values of the Lagrange multipliers and so on. Since the predicted partition is close to the optimal one, this correcting step is not too expensive. In the experiments presented below, except in the vicinity of the end of the path, convergence was reached after a single update.

2.6 Starting Point of the Path

In order to define a starting point for the path, we take benefit of the specificities of Problem 2 when λ goes to infinity. Asymptotically, $H(\lambda)$ is equivalent to $\frac{\lambda}{2} I_m$, so that the contribution of the training examples to the learning problem is restricted to the constraint $y^T \alpha = 0$. The learning problem is "equivalent" to

Problem 4 (Problem equivalent to Problem 2 when λ goes to infinity)

$$\max_{\alpha} \left\{ -\frac{\lambda}{4} \alpha^T \alpha + 1_m^T \alpha \right\}$$

$$s.t. \quad \begin{cases} \forall i \in [\![1, m]\!], \;\; \alpha_i \geqslant 0 \\ y^T \alpha = 0 \end{cases} .$$

In [10] (Appendix C), the authors pointed out that the solution of Problem 4 is given by:

$$\forall i \in [\![1, m]\!], \quad \begin{cases} \alpha_i(\lambda) = \frac{4m_-}{m\lambda} \text{ if } y_i = 1 \\ \alpha_i(\lambda) = \frac{4m_+}{m\lambda} \text{ otherwise} \end{cases} \tag{14}$$

where m_+ is the cardinality of the subset of the training set made up of the examples whose label is 1 and $m_- = m - m_+$. With the values of the Lagrange multipliers at hand, it suffices to compute the limit b_∞ of b_λ as λ goes to infinity to characterize the asymptotic behavior of the classifier. To that end, we make use of the fact that since all the Lagrange multipliers are asymptotically positive, the KT complementary conditions (4) imply that for all i in $[\![1, m]\!]$, $y_i \tilde{h}_\lambda(x_i)$ goes to 1. Then, the combination of (4) and (14) gives

$$b_\infty = \frac{m_+ - m_-}{m} .$$

A direct consequence of this last formula is that if $m_+ \neq m_-$, the limit classifier used in test always returns the category of highest cardinality. Contrary to the case of the ℓ_1 norm, there is no value λ_∞ such that above this value the Lagrange multipliers are constant. All what is known is the asymptotic behavior of the Lagrange multipliers, which is not sufficient to define a starting point in a way similar to that of [22]. However, this knowledge remains useful to derive two criteria: one to check whether a given value is large enough to be the initial value for λ (hereafter noted λ^1), and one to start computing the model selection criterion.

2.7 Overview of the Algorithm and Complexity Analysis

Figure 2 presents the integration of the constitutive elements of the path-following algorithm.

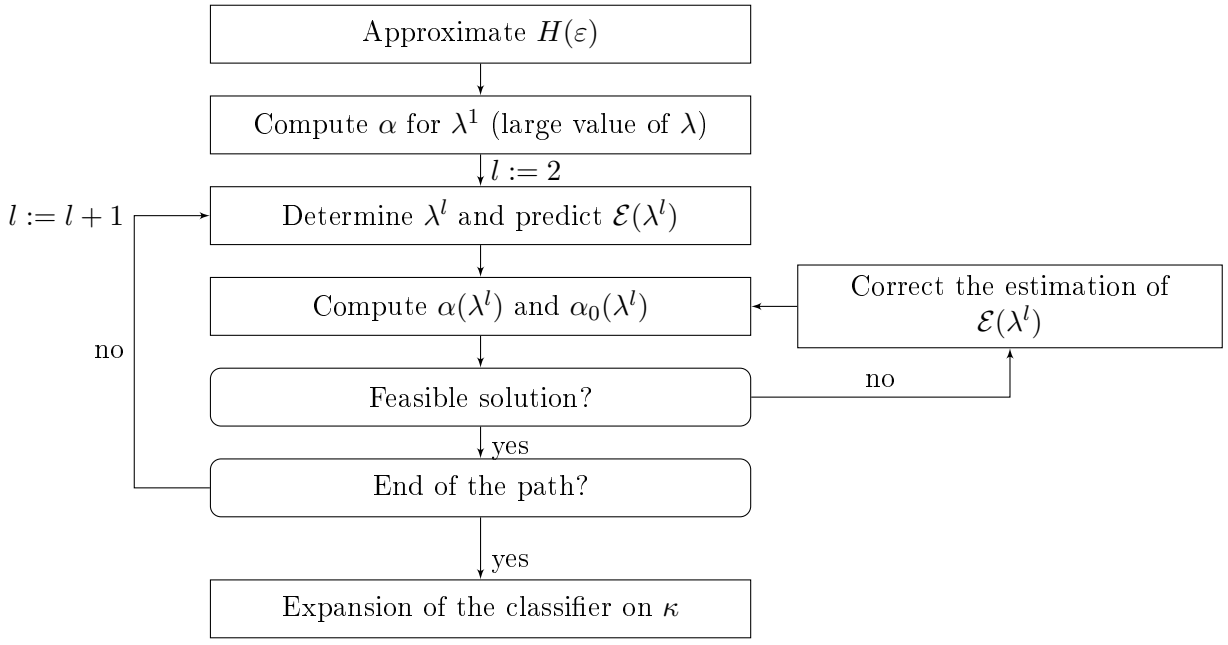

Figure 2: Flowchart of the path-following algorithm.

There are basically five distinct components:

- the low-rank approximation of $H(\varepsilon)$,
- the computation of the Lagrange multipliers,
- the outer loop that generates the sequence $\left(\lambda^l\right)$,
- the inner loop which updates $\mathcal{E}$ (for the current value of λ),

- a post-processing to return the classifier.

We now discuss the complexity of the key parts of the algorithm and some choices regarding their implementation.

2.7.1 Low-Rank Hessian Approximation

The following discussion regards the derivation of $H_r(\varepsilon)$ and the consequences of the approximation. Among all the different algorithms available for matrix factorization are the Incomplete Cholesky Factorization (ICF) proposed in [15], the eigenvalue decomposition truncated to select only the largest eigenvalues, and the density-weighted Nyström method proposed in [33]. Their complexities are respectively in $O\left(mr^2\right)$, $O\left(m^3\right)$, and $O\left(l^3 + lm\right)$ with l the number of landmarks (l can be viewed as a guess of an upper bound on r). Given the results of the comparative study presented in [34], we decided to use Nyström methods. Note that the use of an eigenvalue decomposition (and thus a Nyström decomposition) in our method corresponds to performing a kernel-PCA prior to training a linear SVM with the selected features.

If the RKHS spanned by κ is of finite dimension, the choice of l is straightforward since the value of r is the minimum between this dimension and the number of examples. Otherwise, one cannot expect the matrices $H\left(\varepsilon\right)$ and $H_r\left(\varepsilon\right)$ to be equal, so that the choice for the value of r is harder. It is of major importance as it directly affects the capacity of the class of functions (see [4]). The two main options consist in using a pre-specified fixed rank and selecting every eigenvalue above a given threshold. The fixed-rank method can be connected with the training of a classifier with reduced complexity (see [23]). The use of a threshold on the eigenvalues allows to control the precision of the solution, due to the very nature of the underlying criterion (the spectral norm). For the kernel ridge regression, Proposition 1 of [11] gives an upper bound on the difference of the functions calculated with and without the approximation. When the model is linear (and not affine) in the feature space, this proposition applies to the LS-SVM and thus the ℓ_2-SVM. Such a result allows to gain some insight into the impact of the approximations of the matrix $H(\varepsilon)$ and the regularization coefficient on the classifier. An additional advantage of this technique is that it ensures that $H_r\left(\varepsilon\right)$ is SPD. Thus, we selected the approximation using all the eigenvalues above a given threshold.

It must be borne in mind that using an approximation of $H(\varepsilon)$ does not leave the learning problem unchanged. It induces a change of kernel (all the smaller as the approximation is more accurate). Furthermore, making use of the analytical expression of the new kernel

κ_r is intractable in practice (see the analytical expression of its eigenfunctions in [18]). This raises difficulties which are addressed in Section 2.7.3.

2.7.2 Complexity Analysis of the Outer Loop

As seen in Section 2.5, the computation of λ^{l+1} and the prediction of the corresponding set $\mathcal{E}$ is based on the computation of the derivative of $\alpha_{\mathcal{E}}^a$ with respect to λ. This derivative, given by (13), is obtained by solving a linear system involving the matrix $H_{\mathcal{E},\mathcal{E}}\left(\lambda^l\right)^{-1}$ already handled when computing the values of the dual variables (9). More precisely, the computation of the derivative benefits from the availability of the Cholesky decomposition of $T\left(\lambda^l\right)$. As a consequence, it only takes one forward and one back substitution with several additional matrix multiplications, for a global complexity of $2m_{\mathcal{E}(\lambda^l)}r+2r^2+4m_{\mathcal{E}(\lambda^l)}$ operations. Computing the values of all the λ_i^{l+1} takes $4mr+m$ operations. The overall complexity of the computation of λ^{l+1} is then in $O\left(mr+r^2\right)$.

The complexity for deriving the approximate solution of (9) at λ^{l+1} is in $O\left(m_{\mathcal{E}(\lambda^{l+1})}r+r^3\right)$ once $R_{\mathcal{E}(\lambda^{l+1})}^T R_{\mathcal{E}(\lambda^{l+1})}$ is known. The naive computation of the product $R_{\mathcal{E}(\lambda^{l+1})}^T R_{\mathcal{E}(\lambda^{l+1})}$ requires $m_{\mathcal{E}(\lambda^{l+1})}r^2$ operations making it one of the most expensive steps of the algorithm. In practice, taking benefit of the fact that $\mathcal{E}$ changes slowly, we compute an update of the matrix $R_{\mathcal{E}(\lambda^l)}^T R_{\mathcal{E}(\lambda^l)}$ which only takes $2r^2\Delta\mathcal{E}\left(\lambda^{l+1},\lambda^l\right)$ operations with $\Delta\mathcal{E}\left(\lambda^{l+1},\lambda^l\right) = \left|m_{\mathcal{E}(\lambda^{l+1})}-m_{\mathcal{E}(\lambda^l)}\right|$ the absolute cardinality difference of the two consecutive sets. Given the rule of thumb used to choose λ^{l+1} (see Section 2.5), $\Delta\mathcal{E}\left(\lambda^{l+1},\lambda^l\right)$ can be expected to be close to 2% of $m_{\mathcal{E}(\lambda^l)}$.

2.7.3 Expansion of h_λ in Terms of κ

As stated in Section 2.7.1, the classifier produced by the algorithm is built on the kernel κ_r (whose computation requires $O(mr)$ computations of the value of the kernel κ). In order to obtain a classifier $\hat{h}_\lambda$ which can be efficiently used in test, we propose to switch back from κ_r to κ by taking our inspiration from [4]. Given a subset $\mathcal{J}$ of $[\![1,m]\!]$, this amounts to finding a vector of coefficients $\gamma(\lambda)\in\mathbb{R}^{m_\mathcal{J}}$ so that

$$\hat{h}_\lambda = \sum_{i\in\mathcal{J}} \gamma_i(\lambda)\, y_i \kappa_{x_i} + \alpha_0(\lambda)$$

and $\hat{h}_\lambda$ is identical to the classifier built on κ_r on the training set, leading to

$$\left(y_i \left[\hat{h}_\lambda (x_i) - \alpha_0 (\lambda)\right]\right)_{1 \leqslant i \leqslant m} = \left(y_i \sum_{j=1}^{m_{\mathcal{E}(\lambda)}} \alpha_j (\lambda) y_j \kappa_r (x_j, x_i) \right)_{1 \leqslant i \leqslant m} .$$

This last equation can be reformulated algebraically as follows:

$$H_{\cdot,\mathcal{J}}(\varepsilon)\gamma (\lambda) = R_r R^T_{\mathcal{E}(\lambda)} \alpha_{\mathcal{E}} (\lambda) \quad .$$

In the case of an infinite dimensional feature space, we suggest to choose the set $\mathcal{J}$ equal to $\mathcal{E}(\lambda)$ to preserve the sparsity of the initial SVM. Then, deriving $\gamma (\lambda)$ requires $O\left(m^3_{\mathcal{E}(\lambda)}\right)$ operations. When the dimension of the feature space is r, we can set $m_{\mathcal{J}} = r$ (due to the linear independence of only r rows of H), thus obtaining $\gamma (\lambda)$ in $O\left(r^3\right)$ operations. This result proves to be of particular interest when using polynomial kernels of low degree since it implies that the time needed to evaluate $\hat{h}_\lambda$ on a test example depends on r instead of $m_{\mathcal{E}(\lambda)}$.

3 Model Selection

In this section, we present the criteria of model selection used for our experiments. As pointed out in the introduction, a candidate of choice is the radius-margin bound which is thus presented first. Then, the leave-one-out test error prediction based on the span bound is detailed. We show that this estimate can be obtained as a by-product of the computations of our algorithm fitting the regularization path. This leads us to propose a global model selection procedure integrating this criterion in the regularization path algorithm.

3.1 Radius-Margin Bound

In [31], Vapnik has derived a bound on the leave-one-out cross-validation error of a hard margin SVM.

Theorem 1 (After Sections 10.3 and 10.4 in [31]) *Let us consider a hard margin SVM trained on d_m. Let $\gamma = \left\|\bar{h}\right\|_\kappa^{-1}$ be its margin and R the radius of the smallest ball of $\mathbf{H}_\kappa$*

enclosing the set $\{\Phi(x_i) : 1 \leqslant i \leqslant m\}$. *Then,*

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{y_i h^i(x_i) \leqslant 0\}} \leqslant \frac{4}{m}\frac{R^2}{\gamma^2}$$

where $\mathbb{1}$ *is the standard indicator function and* h^i *is the function computed by the SVM trained on* $d_m \setminus \{(x_i, y_i)\}$.

It must be borne in mind that applying this bound to the ℓ_2-SVM requires to consider the appropriate feature space, i.e., the RKHS induced by κ_λ, which depends on λ. As a consequence, the radius must be computed again for each value of λ considered. This amounts to solving an additional series of convex QP problems.

3.2 Leave-One-Out Test Error Prediction Based on the Span Bound

The *span bound* is an exact bound on the leave-one-out cross-validation error of the ℓ_1-SVM introduced by Vapnik and Chapelle in [32]. It is based on the concept of span of support vectors. We detail here the (leave-one-out) test error prediction derived from the span bound using the hypothesis that the set of support vectors remains the same during the leave-one-out cross-validation procedure. The notation uses explicitly λ to remind that this result holds for both the soft and the hard margin SVM (corresponding to $\lambda \to 0$).

Theorem 2 (Theorem 3 in [32]) *Let us consider an* ℓ_1*-SVM trained on* d_m. *Under the hypothesis that the set of support vectors remains the same during the leave-one-out cross-validation procedure, the following equality holds*

$$\forall i \in [\![1, m]\!], \;\; y_i \left(h_\lambda(x_i) - h_\lambda^i(x_i)\right) = \alpha_i(\lambda) S_i(\lambda)^2$$

with $S_i(\lambda)$ *the distance between* $\Phi(x_i)$ *and the set* $\Lambda_i(\lambda)$ *defined by*

$$\Lambda_i(\lambda) = \left\{ \sum_{j:\alpha_j(\lambda) \in (0,\lambda^{-1}) \wedge j \neq i} \tau_j \Phi(x_j), \sum_{j:\alpha_j(\lambda) \in (0,\lambda^{-1}) \wedge j \neq i} \tau_j = 1 \right\} .$$

This definition of $\Lambda_i(\lambda)$, from [9], is slightly different from the one of [32] but is identical under the hypothesis of invariance of the support vectors.

Corollary 1 (Corollary 1 in [32]) *Under the hypotheses of Theorem 2,*

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}_{\left\{y_i h_\lambda^i(x_i)\leqslant 0\right\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{1}_{\left\{\alpha_i(\lambda)S_i(\lambda)^2 - y_i h_\lambda(x_i)\geqslant 0\right\}}, \tag{15}$$

where the right-hand side of (15) is the aforementioned test error prediction.

When implemented in a naive way, the computation of the test error prediction has a complexity of the same order as that of the leave-one-out (cross-validation) procedure. We now focus on the case of interest in the framework of this study (following the regularization path), the one of the hard margin machine. It is specifically addressed in [9]. In that framework, (15) simplifies into

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}_{\{y_i h^i(x_i)\leqslant 0\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{1}_{\left\{\alpha_i S_i^2 - 1\geqslant 0\right\}} \ .$$

The authors provide a pratical result for obtaining the values of the spans S_i based on the following algebraic reformulation:

$$\forall i \in [\![1,m]\!],\ \ S_i^2 = \min_{\tau}\max_{\mu}\left(\Phi(x_i) - \sum_{j:\alpha_j>0\wedge j\neq i}\tau_j\Phi(x_j)\right)^2 + 2\mu\left(\sum_{j:\alpha_j>0\wedge j\neq i}\tau_j - 1\right)$$

with μ the Lagrange multiplier associated to the constraint $\sum\tau_i = 1$. Let K be the Gram matrix associated to the kernel of the hard margin SVM and $\bar{K}$ the matrix given by:

$$\bar{K} = \begin{pmatrix} K_{\mathcal{E},\mathcal{E}} & 1_{m_\mathcal{E}} \\ 1_{m_\mathcal{E}}^T & 0 \end{pmatrix}$$

with $\mathcal{E}$ being the set of indices of the support vectors. By setting $\bar{\tau} = \left(\tau^T, \mu\right)^T$, the previous min-max problem can be formulated as:

$$S_p^2 = \min_{\tau}\max_{\mu}\left\{\kappa(x_p,x_p) - 2v^T\bar{\tau} + \bar{\tau}^T V\bar{\tau}\right\}$$

with V the submatrix of $\bar{K}$ obtained by removing the row and column of index p and v the p^{th} column of $\bar{K}$ minus its p^{th} component. The existence of V^{-1} in the general case is not discussed here, as only the case of a full rank Gram matrix will be of our concern.

From the optimal value of $\bar{\tau}$ being equal to $V^{-1}v$, Equation 12 of [9] gives the value of S_p:

$$\begin{aligned} S_p^2 &= \kappa\left(x_p, x_p\right) - v^T V^{-1} v \\ &= 1/(\bar{K}^{-1})_{p,p} \ . \end{aligned}$$

The last step comes from the block inversion formula, also called "Woodbury formula" in [9]. Thus the most expensive part of the computation of the leave-one-out test error prediction is the inversion of $\bar{K}$. As this result is only valid for the hard margin machine, in order to apply it to the ℓ_2-SVM, it is required to make use of its hard margin formulation. This implies constructing K from the kernel κ_λ.

3.3 Integration in the Regularization Path Algorithm

This section proposes an efficient implementation of the leave-one-out test error prediction in the framework of the regularization path when the Hessian matrix is of low rank. First we demonstrate a proposition inspired by the method proposed by Cawley and Talbot in [6] for computing the exact leave-one-out (cross-validation) error for the LS-SVM as a by-product of its training. Then from this proposition we present a means to calculate the leave-one-out test error prediction using the low rank property of $H_r\left(\varepsilon\right)$ and thus scaling linearly with $m_{\mathcal{E}}$.

Proposition 4 *The number of errors associated with the leave-one-out test error prediction of the ℓ_2-SVM is equal to that of the leave-one-out cross-validation procedure of the LS-SVM trained on $d_{\mathcal{E}}$ which is*

$$\sum_{i \in \mathcal{E}} \mathbb{1}_{\left\{y_i h_\lambda^i(x_i) \leqslant 0\right\}} = \sum_{i \in \mathcal{E}} \mathbb{1}_{\left\{\frac{\alpha_i(\lambda)}{(A_{\mathcal{E}}(\lambda)^{-1})_{i,i}} - 1 \geqslant 0\right\}} \ . \tag{16}$$

Proof Let $\alpha_0^i(\lambda)$, $\alpha_{\mathcal{E}}^i(\lambda)$ and $\tilde{h}_\lambda^i$ be respectively the bias, the vector of Lagrange multipliers and the function calculated by the LS-SVM trained on $d_{\mathcal{E}} \setminus \{(x_i, y_i)\}$ with the kernel κ_λ. It will be shown later in this proof that $\tilde{h}_\lambda^i\left(x_i\right)$ is equal to $h_\lambda^i\left(x_i\right)$ and thus is the quantity of interest for the leave-one-out procedure.

Cawley and Talbot have shown in [6] that $\tilde{h}_\lambda^i\left(x_i\right)$ can be deduced from quantities involved in the expression of the LS-SVM trained on the whole set d_m. Their demonstration involves the Gram matrix and will be performed here with the Hessian matrix so that it fits in the current framework.

As this computation makes extensive use of (6), let us recall it:

$$\begin{pmatrix} 0 & y_{\mathcal{E}}^T \\ y_{\mathcal{E}} & H_{\mathcal{E},\mathcal{E}} + \frac{\lambda}{2} I_{m_{\mathcal{E}}} \end{pmatrix} \begin{pmatrix} \alpha_0(\lambda) \\ \alpha_{\mathcal{E}}(\lambda) \end{pmatrix} = \begin{pmatrix} 0 \\ 1_{m_{\mathcal{E}}} \end{pmatrix} \quad .$$

For ease of notation, let the coefficients of $A_{\mathcal{E}}(\lambda)$ range from 0 to $m_{\mathcal{E}}$ so that the index of column matches the index of $\alpha_{\mathcal{E}}^a(\lambda)$:

$$A_{\mathcal{E}}(\lambda) = (a_{i,j})_{0 \leqslant i,j \leqslant m_{\mathcal{E}}} \, .$$

From this matrix, we define:

- $A^i \in \mathcal{M}_{m_{\mathcal{E}},m_{\mathcal{E}}}(\mathbb{R})$ the matrix $A_{\mathcal{E}}(\lambda)$ with the row and the column of index i removed,
- $a_i \in \mathbb{R}^{m_{\mathcal{E}}}$ the column of index i with its coefficient of index i removed,
- $a_{i,i} \in \mathbb{R}$ the coefficient of indexes (i,i).

With these notations at hand, the row of index i ($i \in [\![1, m_{\mathcal{E}}]\!]$) of System (6), corresponding to the Kuhn-Tucker optimality condition for the example i, can be rewritten as

$$a_i^T \left(\alpha_0(\lambda) \alpha_{\mathcal{E}\setminus\{i\}}(\lambda)^T\right)^T = 1 - a_{i,i}\alpha_i(\lambda) \tag{17}$$

while the other rows are

$$\left(A^i \; a_i\right) \left(\alpha_0(\lambda) \alpha_{\mathcal{E}\setminus\{i\}}(\lambda)^T \alpha_i(\lambda)\right)^T = (0 \; 1_{m_{\mathcal{E}}-1}^T)^T \quad . \tag{18}$$

This decomposition of $A_{\mathcal{E}}(\lambda)$ allows to express the learning problem of the LS-SVM trained on $\mathcal{E} \setminus \{i\}$ in terms of A^i:

$$A_i \begin{pmatrix} \alpha_0^i(\lambda) \\ \alpha_{\mathcal{E}}^i(\lambda) \end{pmatrix} = \begin{pmatrix} 0 \\ 1_{m_{\mathcal{E}}-1} \end{pmatrix} . \tag{19}$$

From the definition of a_i we have $y_i \tilde{h}_\lambda^i(x_i) = H_{i,\mathcal{E}}(\lambda)\alpha_{\mathcal{E}}^i(\lambda) + y_i\alpha_0^i(\lambda) = a_i^T \left(\alpha_0^i(\lambda) \; \alpha_{\mathcal{E}}^i(\lambda)^T\right)^T$. By substituting (19) in this expression we get:

$$y_i \tilde{h}_\lambda^i(x_i) = a_i^T \left(A^i\right)^{-1} \left(0 \; 1_{m_{\mathcal{E}}-1}^T\right)^T \quad .$$

Equations (17) and (18) allow to simplify this formula to express $y_i\tilde{h}^i_\lambda(x_i)$ as a function of $\alpha_i\,(\lambda)$:

$$\begin{aligned} y_i\tilde{h}^i_\lambda(x_i) &= a_i^T \left(A^i\right)^{-1} \left(A^i\ a_i\right) \left(\alpha_0(\lambda)\ \alpha_{\mathcal{E}\setminus\{i\}}(\lambda)^T\alpha_i(\lambda)\right)^T \\ &= a_i^T \left(A^i\right)^{-1} A^i \left(\alpha_0(\lambda)\alpha_{\mathcal{E}\setminus\{i\}}(\lambda)^T\right)^T + \alpha_i(\lambda)a_i^T \left(A^i\right)^{-1} a_i \\ &= 1 - \alpha_i(\lambda)a_{i,i} + \alpha_i(\lambda)a_i^T \left(A^i\right)^{-1} a_i \\ &= 1 - \alpha_i(\lambda)\left(a_{i,i} - a_i^T \left(A^i\right)^{-1} a_i\right) \\ &= 1 - \frac{\alpha_i(\lambda)}{(A_\mathcal{E}(\lambda)^{-1})_{i,i}}\ . \end{aligned} \tag{20}$$

The last step of (20) is obtained thanks to the application of the block inversion formula. It is noteworthy to point out that (20) is identical to Equation (12) of [6]. As previously mentioned, the outputs of the hard margin classifier and the soft margin classifier for the example i are equal:

$$\begin{aligned} \tilde{h}^i_\lambda(x_i) &= \sum_{j\neq i}\left(\alpha_j(\lambda)y_j\kappa(x_i,x_j) + \delta_{i,j}\frac{\lambda}{2}\right) + \alpha_0(\lambda) \\ &= \sum_{j\neq i}\alpha_j(\lambda)y_j\kappa(x_i,x_j) + \alpha_0(\lambda) \\ &= h^i_\lambda(x_i)\ . \end{aligned} \tag{21}$$

Combining Equations (20) and (21) gives the number of misclassified examples of $d_\mathcal{E}$ for the LS-SVM trained on $d_\mathcal{E}$. As only the examples of $d_\mathcal{E}$ are of interest, the number of errors of (15) and (16) are exactly the same, which concludes the proof. ■

This proposition is twofold as it ensures good model selection properties due to the well established leave-one-out test error prediction based on the spans and it allows to compute it efficiently. As previously seen, the matrix $A_\mathcal{E}(\lambda)$ plays a central role for model selection. The block inversion formula gives the elements of the diagonal of the inverse of $A_\mathcal{E}(\lambda)$:

$$\begin{aligned} A_\mathcal{E}(\lambda)^{-1} &= \begin{pmatrix} s^{-1} & -s^{-1}y_\mathcal{E}^T H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} \\ -H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1}y_\mathcal{E}s^{-1} & H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} + H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1}y_\mathcal{E}s^{-1}y_\mathcal{E}^T H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{s} & -\frac{1}{s}\nu^T \\ -\frac{1}{s}\nu & H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} + \frac{1}{s}\nu\nu^T \end{pmatrix} \end{aligned}$$

with $s = -y_\mathcal{E}^T H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1}y_\mathcal{E} = -y_\mathcal{E}^T\nu$.

For $i \in [\![1, m_{\mathcal{E}}]\!]$, the elements of the diagonal are:

$$A_{\mathcal{E}}(\lambda)^{-1}_{i,i} = (H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1})_{i,i} + \frac{1}{s}\nu_i^2 \ . \tag{22}$$

Thanks to the low-rank approximation of $H_{\mathcal{E},\mathcal{E}}(\varepsilon)$, the computational complexity of the diagonal elements of the inverse matrix is linear in $m_{\mathcal{E}}$. Hereafter comes the detail of this proposition. Keeping the notations of Section 2.4, we obtain

$$\begin{aligned} H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1} \simeq \left(H_{\mathcal{E},\mathcal{E},r}\left(\varepsilon\right) + \frac{\lambda-\varepsilon}{2} I_{m_{\mathcal{E}}}\right)^{-1} &= \frac{2}{\lambda-\varepsilon} I_{m_{\mathcal{E}}} - \frac{4}{(\lambda-\varepsilon)^2} R_{\mathcal{E}} T\left(\lambda\right)^{-1} R_{\mathcal{E}}^T \\ &= \frac{2}{\lambda-\varepsilon} I_{m_{\mathcal{E}}} - \frac{4}{(\lambda-\varepsilon)^2} (L_{\mathcal{E}}^{-1} R_{\mathcal{E}}^T)^T (L_{\mathcal{E}}^{-1} R_{\mathcal{E}}^T). \end{aligned}$$

Computing the matrix $L_{\mathcal{E}}^{-1} R_r^T$ yields a complexity in $O(m_{\mathcal{E}} r^2)$ operations. Let $u_{i,j}$ be its general term. The diagonal terms of $H_{\mathcal{E},\mathcal{E}}(\lambda)^{-1}$ are given by

$$\forall i \in [\![1, m_{\mathcal{E}}]\!], \ \ (H_{\mathcal{E},\mathcal{E},r}(\lambda,\varepsilon)^{-1})_{i,i} = \frac{2}{\lambda-\varepsilon} - \frac{4}{(\lambda-\varepsilon)^2} \sum_{j=1}^{r} u_{j,i}^2 \ . \tag{23}$$

Combining (22) and (23), and using the fact that ν and $L_{\mathcal{E}}$ are already precomputed, allows to obtain a complexity in only $O(m_{\mathcal{E}} r^2)$ operations for calculating the leave-one-out test error prediction. When r is small compared to m, this complexity is inferior to that of the QP problem solved to derive the radius.

The integration of the model selection in the regularization path algorithm results in Algorithm 1. The main differences are the computation of the leave-one-out test error prediction and the expansion of the optimal (with respect to the test error prediction) classifier on κ.

4 Experimental Results

We now present results from various numerical simulations. First a comparison of the solution obtained by our path-following algorithm and a state-of-the-art algorithm is performed. Then we illustrate the good behavior of the leave-one-out test error prediction. The central experiment is the comparison of our model selection procedure with another state-of-the-art method on severals data sets. We conclude with a comparison of the computation time of our model selection procedure and a training algorithm dedicated to large data sets.

Algorithm 1 Algorithm of the model selection procedure using the regularization path

Require: $H_r(\varepsilon) = RR^T$, $\lambda_{\max} > \lambda_{\min} > 0$.

$\lambda \leftarrow \lambda_{\max}$
$l \leftarrow 1$
$[\alpha(\lambda^l), \alpha_0(\lambda^l), \nu, \rho, L] \leftarrow$ lagrange_multipliers_calculation$(R, \mathcal{E}(\lambda^l), y, \lambda^l)$
while $\lambda^l > \lambda_{\min}$ **do**
  $l \leftarrow l + 1$
  $[\lambda^l, \mathcal{E}(\lambda^l)] \leftarrow$ next_change_prediction$(\alpha(\lambda^l), \alpha_0(\lambda^l), \nu, \rho, L, R, \lambda^l)$
  repeat
    $[\alpha(\lambda^l), \alpha_0(\lambda^l), \nu, \rho, L] \leftarrow$ lagrange_multipliers_calculation$(R, \mathcal{E}(\lambda^l), y, \lambda^l)$
    corrected_$\mathcal{E}(\lambda^l) \leftarrow$ update_set$(\alpha(\lambda^l), \alpha_0(\lambda^l), R, \lambda^l)$
  until corrected_$\mathcal{E}(\lambda^l) == \mathcal{E}(\lambda^l)$
  $loo\_approx^l \leftarrow$ test_error_prediction$(\alpha(\lambda^l), \lambda^l, R, \mathcal{E}, \nu, \rho, L)$
end while
$\lambda^{\text{opt}} = \text{argmin}_l \, loo\_approx^l$
$\gamma \leftarrow$ classifier_expansion$(R, \mathcal{E}(\lambda^{\text{opt}}), \alpha(\lambda^{\text{opt}}))$
return $\gamma, \alpha_0(\lambda^{\text{opt}})$

4.1 Setup

All Hessian matrix approximations involve a Nyström decomposition method. When the training set is smaller than 4000 examples, a density-weighted Nyström method is used, otherwise, a uniform sampling method is applied. Each data set is standardized.

Table 1 shows some statistics about the data sets. Most of them are from the Rätsch database (`http://www.raetschlab.org/Members/raetsch/benchmark`). The Spam data set is taken from the UCI repository [17] while ijcnn1 and a9a are taken from Lin's homepage (`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`). The artificial data set will be detailed later.

Unless otherwise specified, a Gaussian kernel (whose bandwidth will be denoted σ) is used. To avoid numerical instabilities, ε is set equal to 10^{-8}. The regularization coefficient ranges from 10^{-6} to 10^{7}. The leave-one-out test error prediction is evaluated for each λ^l. All the computational times include the approximation of $H(\varepsilon)$, the path-following itself, the evaluations of the leave-one-out test error prediction and the expansion of the optimal classifier (when required). All our codes are written in MATLAB (R2010B) and were executed on a PC using 2-XeonE5540 (8 x 2.53 GHz) processors and 32GB of RAM memory.

Data set	#features	#training	#test	#realizations
banana	2	400	4900	100
breast cancer	9	200	77	100
diabetis	8	468	300	100
flaresolar	9	666	400	100
german	20	700	300	100
heart	13	170	100	100
image	18	1300	1010	20
ringnorm	20	400	7000	20
splice	60	1000	2175	20
tyroid	5	140	75	100
titanic	3	150	2051	100
twonorm	20	400	7000	100
waveform	21	400	4600	100
Spam	57	3601	1000	30
ijcnn1	22	49990	91701	1
a9a	123	32561	16281	1

Table 1: Statistics on the data sets. #realizations corresponds to the number of realizations of training sets and test sets for each data set.

4.2 Optimality of the Solution Obtained by the Regularization Path

This section illustrates the fact that the low-rank approximation can lead to the same result as the classical approach. A state-of-the-art algorithm, the ℓ_2 version of libsvm (ℓ_2-libsvm), is used for comparison on three data sets with different kernels. Since a good approximation is required, the following settings are chosen:

- the threshold on the eigenvalues is set to 10^{-6},
- 80% of the training set is used for building the approximation.

Good quality of the first eigenvalues and eigenvectors estimations is ensured by performing the density-weighted Nyström approximation on a large part of the training set. Table 2 shows that the solutions found by our path-following method and ℓ_2-libsvm are close. It is important to note that ℓ_2-libsvm has difficulties to converge when the bandwidth of the Gaussian kernel is large, thus explaining why its training time is so high. For ℓ_2-libsvm, we display ten times the average value in the table, assuming that when selecting model by means of a grid, one would use 10 different values of λ.

4.3 Comparison of the Model Selection Criteria

The relative behavior of the leave-one-out cross-validation bounds (or test error estimators) has already been extensively studied in the literature (for example in [9, 10]). Two main

Data set	Kernel (σ)	Rank $H_r(\varepsilon)$	Mean time in seconds (number of training) Path	ℓ_2-libsvm	Ratio of obj. functions
banana	Linear (X)	2	0.5996 (44)	1.6229 (10)	0.99988
	Gaussian (0.2)	317	1.5592 (49)	0.8799 (10)	0.98236
	Gaussian (0.6)	147	0.95875 (45)	1.455 (10)	0.99993
	Gaussian (1)	81	0.84712 (46)	16.0839 (10)	0.99945
	Gaussian (1.4)	56	0.79836 (45)	18.8339 (10)	1.0001
	Gaussian (1.8)	44	0.8895 (44)	13.0951 (10)	0.9991
twonorm	Linear (X)	20	0.69215 (50)	0.99338 (10)	0.99963
	Gaussian (10)	319	1.0794 (49)	1.1886 (10)	0.99304
	Gaussian (15)	319	1.0554 (49)	1.1857 (10)	0.99361
	Gaussian (20)	319	1.0842 (49)	1.1932 (10)	0.99231
	Gaussian (25)	319	1.0724 (49)	1.1952 (10)	0.99305
	Gaussian (30)	289	0.99932 (49)	1.2005 (10)	0.99553
	Gaussian (35)	244	0.90694 (49)	1.2022 (10)	0.99558
	Gaussian (40)	231	0.90001 (49)	1.2066 (10)	0.99672
image	Linear (X)	14	2.267 (44)	33.7736 (10)	1.0001
	Gaussian (2)	905	9.3627 (47)	5.4173 (10)	0.98943
	Gaussian (3)	873	7.8396 (46)	5.2021 (10)	0.99386
	Gaussian (4)	778	7.7303 (45)	5.5274 (10)	0.99759
	Gaussian (5)	683	6.0358 (45)	6.3224 (10)	0.99899
	Gaussian (6)	598	5.3664 (41)	5.6593 (10)	0.99977
	Gaussian (7)	527	5.3114 (41)	5.7693 (10)	0.99987

Table 2: Comparison of the values of the objective function evaluated with the Lagrange multipliers obtained from ℓ_2-libsvm and with our path-following algorithm. For our path-following algorithm, the number in parentheses corresponds to the number of training performed. For ℓ_2-libsvm, the time corresponds to ten times the average (over all the values of λ considered) of the training time.

conclusions can be drawn: the minima of both the radius-margin bound and the leave-one-out test error prediction are adequate criteria for model selection and the leave-one-out test error prediction is an accurate estimator of the generalization error. Figure 3 illustrates these properties on the banana data set.

4.4 Accuracy in Terms of Model Selection

This section illustrates the use of a low-rank approximation of the Hessian matrix for computing the regularization path to perform model selection. The algorithm of reference for model selection, introduced in [9], is based on the gradient descent using differentiable spans. Three experiments are performed with data sets of increasing size. Here are the details of the comparison procedure:

- For each realization of the data set, the optimal classifier computed by the algorithm on the training set is tested on the corresponding test set. The mean value and the standard deviation over all the realizations are given in the tables.

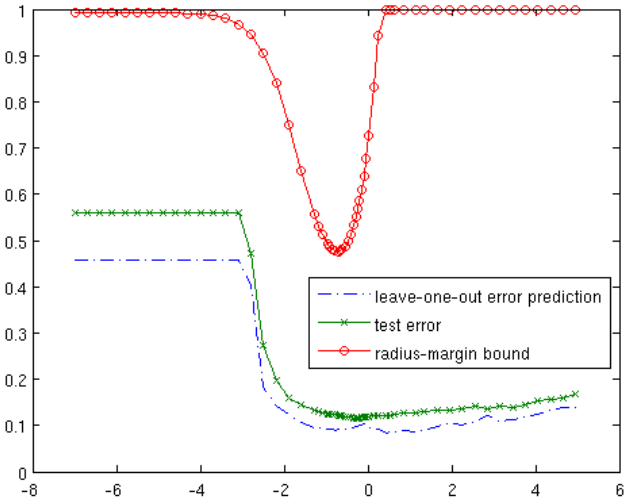

Figure 3: Evolution of the radius-margin bound (truncated to 1), the leave-one-out test error prediction and the test error along the regularization path.

- A resampled paired Student t test (described in Section 3.3 of [13]) is performed on the frequency of test errors. If a difference is statistically significant, the best result is written in bold.

The Rätsch database The Rätsch database has been extensively used as benchmark for binary classification (see for example [26, 14, 23, 6]). In this trial, due to the small size of the data sets, the density-weighted Nyström decomposition is computed by using 60% of the training set as landmarks and only keeping the eigenvalues greater than 10^{-4}.

Table 3 presents comparative results based on the σ selected by the gradient method while Table 4 corresponds to the heuristic proposed in [34]. This method sets σ equal to the square root of the mean distance between each point and the center of mass of the training set. Even with this simple heuristic, the performance of our algorithm matches the one of Chapelle's algorithm. The recognition rate is on par with the literature (see for example [10]). These two tables show that both methods provide good performance in terms of model selection when no assumption on the rank needed to approximate $H(\varepsilon)$ is made.

The data sets of the Rätsch database have a relatively large number of features for a small number of training examples. Thus the rank needed to obtain good performance on this database is not very small compared to the number of examples, explaining the similarity of the computation times.

Data set	Test error in % Mean (standard deviation)		Time in seconds	
	Path	Gradient	Path	Gradient
banana	11.07 (0.88)	**10.85** (0.73)	0.6319	**0.5612**
breast cancer	27.10 (4.67)	26.64 (4.56)	0.4613	**0.2551**
diabetis	24.04 (1.94)	**23.81** (1.94)	**0.8224**	1.2781
flaresolar	**34.49** (1.87)	34.84 (1.82)	**0.4011**	2.4215
german	23.77 (2.13)	23.65 (2.03)	**1.9661**	2.8821
heart	16.83 (3.47)	16.71 (3.11)	0.4741	**0.1586**
image	**03.32** (0.72)	03.90 (0.69)	**6.5991**	8.1389
ringnorm	01.69 (0.40)	**01.61** (0.15)	0.9900	**0.4171**
splice	11.87 (0.72)	**11.16** (0.70)	**4.0894**	5.8144
tyroid	07.08 (3.69)	07.01 (3.66)	**0.4575**	0.0728
titanic	23.05 (1.11)	**22.59** (0.88)	**0.3804**	0.1292
twonorm	02.67 (0.34)	02.69 (0.18)	0.9998	**0.4748**
waveform	10.09 (0.56)	**09.90** (0.39)	1.0605	**0.7644**

Table 3: Performance on the Rätsch database with the value of σ given by Chapelle's algorithm.

The Spam data set The Spam data set is much larger than the data sets of the Rätsch database. This increase in the number of examples allows us to show that our path-following method for model selection scales well with the number of examples. As there is no predefined training and test sets, for each realization 1000 test examples are randomly chosen from the database while keeping the rest for the training set.

The choice of the number of eigenvectors retained (600) is based on the classical "elbow" criterion used for eigenvector selection in PCA. Table 5 shows that although the approximation is loose, the recognition rate is still good.

A synthetic case: Large training set in low dimension In order to evaluate the scalability of the algorithm in a favourable case, we built an artificial data set in $\mathbb{R}^2$. The examples belonging to the positive category are drawn from a Gaussian distribution while the negative ones are drawn from a mixture of Gaussians, both categories being equiprobable.

$$p(x|y=+1) = p_{\mathcal{N}}(\mu^+, \Sigma^+, x)$$
$$p(x|y=-1) = \frac{1}{2}p_{\mathcal{N}}(\mu_1^-, \Sigma_1^-, x) + \frac{1}{2}p_{\mathcal{N}}(\mu_2^-, \Sigma_2^-, x)$$

where

$$p_{\mathcal{N}}(\mu, \Sigma, x) = \frac{1}{2\pi \det(\Sigma)^{1/2}} \exp^{-\frac{1}{2}(x-\mu)^T(\Sigma)^{-1}(x-\mu)},$$

Data set	Test error in % Mean (standard deviation)		Time in seconds	
	Path	Gradient	Path	Gradient
banana	11.24 (0.95)	**10.85** (0.73)	0.7149	**0.7017**
breastcancer	27.35 (4.22)	**26.64** (4.56)	0.5152	**0.3171**
diabetis	24.05 (2.03)	**23.81** (1.94)	**1.1452**	1.6742
flaresolar	**34.40** (1.99)	34.84 (1.82)	**0.4521**	3.1518
german	23.76 (2.19)	23.65 (2.03)	**2.2267**	3.3662
heart	17.30 (3.51)	**16.71** (3.11)	0.5602	**0.2204**
image	**03.24** (0.74)	03.90 (0.69)	**2.8286**	9.1764
ringnorm	02.08 (0.37)	**01.61** (0.15)	0.9758	**0.4092**
splice	11.67 (0.74)	**11.16** (0.70)	**4.7143**	6.9180
thyroid	06.71 (3.28)	07.01 (3.66)	0.5089	**0.1000**
titanic	23.45 (4.28)	**22.59** (0.88)	0.3827	**0.1586**
twonorm	02.68 (0.34)	02.69 (0.18)	0.9846	**0.4683**
waveform	10.29 (0.75)	**09.90** (0.39)	0.9157	**0.6260**

Table 4: Performance on the Rätsch database with σ set with the heuristic of [34].

Criteron	Test error in % Mean (standard deviation)		Time in seconds	
	Path	Gradient	Path	Gradient
default	**6.22** (0.87)	6.58 (0.84)	**27.8510**	97.5712
rank=600	6.64 (0.97)	6.56 (0.76)	**15.9084**	102.9281

Table 5: Performance on the Spam data set with σ computed using the heuristic of [34]. The line "default" corresponds to a Nyström approximation using 60% of the training set while "rank=600" uses 600 landmarks (the approximation of the Hessian matrix can be at most of rank 600). Both experiments use an eigenvalue threshold of 10^{-3}.

$$\mu^+ = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \Sigma^+ = \begin{pmatrix} 8 & -6 \\ -6 & 8 \end{pmatrix}, \mu_1^- = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_1^- = \begin{pmatrix} 1/4 & 0 \\ 0 & 1/4 \end{pmatrix}, \mu_2^- = \begin{pmatrix} 5 \\ -2 \end{pmatrix}, \Sigma_2^- = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

A Monte Carlo estimate of the Bayes error is 12.11%. Ten realizations of the training set have been built for each selected value of m (see Table 6). The evaluation of the generalization error of each classifier is based on a large test set (30000 examples).

Table 6 illustrates that both model selection procedures achieve similar performance. The gain in computation time is clearly due to the low-rank approximation. It is noteworthy to recall that Chapelle's algorithm is not designed to handle large data sets. Therefore, the following section compares our algorithm to a training algorithm dedicated to this kind of problem.

m	Test error(%)		Time(s)		Rank
	Path	Gradient	Path	Gradient	
1000	14.22 (0.28)	14.05 (0.25)	5.68	9.21	127.5
2000	13.97 (0.16)	**13.93** (0.13)	12.84	57.55	165.8
3000	13.86 (0.17)	13.85 (0.19)	20.87	101.1	181.4
4000	13.85 (0.33)	13.83 (0.33)	28.41	262.56	155.2
5000	13.84 (0.19)	13.82 (0.2)	37.98	446.56	154.7

Table 6: Performance on the artificial data set with the value of σ obtained by Chapelle's algorithm.

		ijcnn1	a9a
	Kernel	$(32\langle x_1, x_2\rangle + 1)^2$	$(0.032\langle x_1, x_2\rangle + 1)^2$
ℓ_2-liblinear/ℓ_2-libsvm	C	0.125	8
	Termination criteron	10^{-6}	10^{-6}
Path	Number of landmarks	500	1500
	Threshold	10^{-6}	10^{-6}

Table 7: Parameterization of the algorithms for the comparison. Due to the size of the data sets, the approximation method is the uniform Nyström method. The value of C corresponds to the optimal value provided by the grid search.

4.5 Performance and Training Time Comparison on Large Data Sets

Obviously, the low-rank approximation is well suited for a low-rank Hessian matrix. A low-degree polynomial kernel can induce such a property. To the best of our knowledge, there is no model selection procedure devised for this case. Thus, two training algorithms dedicated to these kernels are used for comparison: the modified versions of ℓ_2-liblinear and ℓ_2-libsvm intoduced in [7] (liblinear is known as one of the fastest solvers for linear SVMs). For both of them, model selection results from a 5-fold cross-validation implemented over a grid search. Proceeding as in [7] (preprint), we chose the following range: $C = \lambda^{-1} \in \left\{2^{-3}, 2^{-1}, \dots 2^{7}, 2^{9}\right\}$.

Data set	Algorithm	Training time in seconds	Testing time in seconds	Test error in %	Rank
ijcnn1	Path	**30.2s**	1.9s	2.44%	208
	ℓ_2-liblinear	57.8s	0.4s	2.46%	-
	ℓ_2-libsvm	$>$ 24 hours	27s	2.46%	-
a9a	Path	**179.9s**	1.9s	14.8%	1178
	ℓ_2-liblinear	459.7s	0.05s	14.7%	-
	ℓ_2-libsvm	46803s	31s	15.2%	-

Table 8: Training times and testing times for ijcnn1 and a9a data sets with a degree 2 polynomial kernel.

The parameterization of the algorithm is presented in Table 7 while the result of the comparison is in Table 8. This last table shows a significant reduction of the training time. Furthermore, the expansion of the classifier on r examples allows to keep the testing time small. These results are all the more interesting as our procedure is not restricted to polynomial kernels of degree two but can also be used with task specific kernels. Therefore, the use of our procedure is very appealing in this framework.

5 Conclusions and Ongoing Research

In this article, a path-following algorithm for setting the value of the regularization coefficient of the ℓ_2-SVM has been introduced. It relies on two original contributions: an algorithm following the regularization path and the integration in this algorithm of the leave-one-out test error prediction based on the well-known Span bound. The main advantage of this combination is to avoid local minima of the objective function (since it covers the whole parameter space). Furthermore, this method is particularly efficient when the rank of the Gram matrix is small. Indeed, the complexity of model selection is linear in the number of examples when the rank of the Gram matrix does not depend on the number of examples. In this case, we present experimental evidence that the gain in time is significant.

The main practical difficulty raised by the implementation lies in the adequate choice of the rank when resorting to approximation. Automatic methods for selecting it are currently under investigation. We favour two main options. The first one uses the work of Girolami [18] dealing with kernel-PCA, the second one relies on a variable rank for the approximation. Our final objective is to extend this model selection algorithm to the multi-class case, more precisely to our quadratic loss multi-class SVM, the M-SVM^2 [21].

Acknowledgments

This work was funded by the Fédération Charles Hermite and the Région Lorraine.

References

[1] E.L. Allgower and K. Georg. Continuation and path following. *Acta Numerica*, 2:1–64, 1993.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[3] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, Boston, 2004.

[4] G. Blanchard, P. Massart, R. Vert, and L. Zwald. Kernel projection machine: a new tool for pattern recognition. In *NIPS 17*, pages 1649–1656, 2005.

[5] P. Burman. A comparative study of ordinary cross-validation, ν-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76(3):503–514, 1989.

[6] G.C. Cawley and N.L.C. Talbot. Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, 2007.

[7] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11:1471–1490, 2010.

[8] O. Chapelle. Training a support vector machine in the primal. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*, chapter 2, pages 29–50. The MIT Press, Cambridge, MA, 2007.

[9] O. Chapelle, V.N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[10] K.-M. Chung, W.-C. Kao, C.-L. Sun, L.-L. Wang, and C.-J. Lin. Radius margin bounds for support vector machines with the RBF kernel. *Neural Computation*, 15(11):2643–2681, 2003.

[11] C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *AISTATS 2010*, pages 113–120, 2010.

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

[13] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[14] K. Duan, S.S. Keerthi, and A.N. Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59, 2003.

[15] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

[16] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Chichester, second edition, 1987.

[17] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[18] M. Girolami. Orthogonal series density estimation and the kernel eigenvalue problem. *Neural Computation*, 14(13):669–688, 2002.

[19] C. Gold and P. Sollich. Model selection for support vector machine classification. *Neurocomputing*, 55(1-2):221–249, 2003.

[20] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[21] Y. Guermeur and E. Monfrini. A quadratic loss multi-class SVM for which a radius-margin bound applies. *Informatica*, 22(1):73–96, 2011.

[22] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[23] S.S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

[24] S.S. Keerthi, V. Sindhwani, and O. Chapelle. An efficient method for gradient-based adaptation of hyperparameters in SVM models. In *NIPS 19*, pages 673–380, 2007.

[25] A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*, 3, 1969. (in Russian).

[26] G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[27] S. Rosset. Following curved regularized optimization solution paths. In *NIPS 17*, pages 1153–1160, 2005.

[28] M. Stone. Asymptotics for and against cross-validation. *Biometrika*, 64(1):29–35, 1977.

[29] J.A.K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle. Least squares support vector machine classifiers: a large scale algorithm. *Proceeding of the European Conference on Circuit Theory and Design*, pages 839–842, 1999.

[30] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[31] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.

[32] V.N. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000.

[33] K. Zhang and J.T. Kwok. Density-weighted Nyström method for computing large kernel eigensystems. *Neural Computation*, 21(1):121–146, 2009.

[34] K. Zhang, I.W. Tsang, and J.T. Kwok. Improved Nyström low-rank approximation and error analysis. In *ICML'08*, pages 1232–1239, 2008.