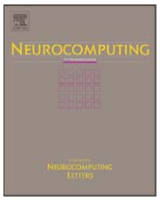
Neurocomputing 73 (2010) 1018–1023

Contents lists available at ScienceDirect

Neurocomputing

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



Letters

Fast error estimation for efficient support vector machine growing[☆]

A. Navia-Vázquez [*], R. Díaz-Morales

DTSC, Univ. Carlos III de Madrid, Avda Universidad 30, 28911-Leganés, Madrid, Spain

A R T I C L E I N F O

Article history:
Received 21 July 2009
Received in revised form
9 December 2009
Accepted 11 December 2009
Communicated by H. Yu
Available online 22 December 2009

Keywords:
Fast error estimation
Growing
Semiparametric
Support vector machine

A B S T R A C T

Support vector machines (SVMs) have become an off-the-shelf solution to solve many machine learning tasks but, unfortunately, the size of the resulting machines is quite often exceedingly large, which hampers their use in those practical applications demanding extremely fast response. Some methods exist to prune the models after training, but a full SVM model needs to be trained first, which usually represents a large computational cost. Furthermore, the reduction algorithms are prone to fall in local minima and also represent an additional non-negligible computational cost. Alternative procedures based on incrementally growing a semiparametric model provide a good compromise between complexity, machine size and performance. We investigate here the potential benefits of a fast error estimation (FEE) mechanism to improve the semiparametric SVM growing process. Precisely, we propose to use the FEE method to identify the best node to be added to the model in every growing step, by selecting the candidate with the lowest cross-validation error. We evaluate the proposed approach by evaluating the performance of the algorithm in benchmarks with real world datasets from the UCI machine learning repository.

1. Introduction: the machine size problem

Support vector machines (SVMs) have gained a lot of popularity because of their good performance in many real world problems and their ability to automatically adjust the machine size for a given problem once the hyperparameters are set [12,17]. The resulting machine size (equal to the number of support vectors) is usually exceedingly high, hampering the use of such models in many real world applications. Some approaches propose to reduce the machine size after training a full model, the most successful methods follow the "reduced-set" procedure [2] (or analogous methods [13,8,14]), but they have to start from a full SVM solution, and then solve a pre-image problem. The procedure for computing pre-images, as analyzed in [1,3], is rather cumbersome, and prone to fall in local minima. Although the authors in [16] propose and improved pre-image computation method, they need to incorporate a genetic algorithm for carrying out those improved minimizations, which complicates the operation of the method. Alternative approaches, known as semiparametric support vector machines (SSVMs) [9,7], propose to incrementally grow the machine architecture. The direct or "brute-force" approach for selecting the best node to be added demands a lot of computation, since too many SSVM models need to be trained. Approximate mechanisms are therefore needed. In the past, we have explored the use of a heuristical criterion based on kernel projections to choose the best candidate [9] and a modified sparse greedy matrix approximation (SGMA) mechanism to select the candidate that obtains the largest error reduction in the representation of the support vectors [7]. We will refer to the latter as SGMA-SVM in what follows. In these two approaches only information from the input space is used, the class labels are not taken into consideration.

In this work, we propose to take advantage of the class labels using a fast error estimation (FEE) mechanism to compute the classification error that we obtain if we add a particular node to the SSVM model, of course, without needing to train all those models. We can then select the node that will mostly reduce the cross-validation classification error in every growing step. The rest of the paper is structured as follows. In Section 2 we revisit the formulation of SSVMs as well as their training algorithm. In Section 3 we derive the fast error estimation method and use it in the growing stage of the SVM training, to obtain the FEE-SVM algorithm. In Section 4 we illustrate the achieved performance using real world datasets and, finally, we close the paper in Section 5 with some conclusions and further work.

2. Semiparametric SVMs for classification revisited

Semiparametric support vector machines [6,9,7] are a means to control the size of the classifier by introducing a predefined

[☆] This work has been partially supported by Spain Government under Grant TEC2008-02473.

[*] Corresponding author. Tel.: +34 916245977; fax: +34 916248749.
E-mail address: navia@tsc.uc3m.es (A. Navia-Vázquez).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.12.008

model in the formulation of the SVM problem. Consider a binary classification problem defined by a set of labelled input patterns $\{\mathbf{x}_i, y_i\}_{i=1}^P$ with $\mathbf{x}_i \in R^N$ and $y_i \in \{-1,+1\}$. The SVM first projects the input data onto a high dimensional space, F, by means of a nonlinear projection $\boldsymbol{\phi}(\cdot)$ in a way that inner products between projected vectors can be computed by means of a kernel function $k(\mathbf{x}_i,\mathbf{x}_j)=\boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_j)$ (i.e., the "kernel trick"). Then the SVM finds a maximal margin linear classifier in F (optimal hyperplane, OH, defined by $\mathbf{w}$ and b), $f(\mathbf{x})=sign(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})+b)$, such that $\{\mathbf{w},b\}$ is the solution to

$$\min_{\mathbf{w},b,\xi_i}\left\{\frac{1}{2}\mathbf{w}^T\mathbf{w}+C\sum_{i=1}^{P}\xi_i\right\} \tag{1}$$

subject to

$$y_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)+b)-1+\xi_i\ge 0, \qquad \forall i=1,\ldots,P \tag{2}$$

$$\xi_i\ge 0, \qquad \forall i=1,\ldots,P \tag{3}$$

where ξ_i are positive slack variables introduced to deal with non-separable problems and C is the penalty for patterns incorrectly classified or inside the margin. After introducing the restrictions in the functional by means of Lagrange multipliers and applying the Karush–Kuhn–Tucker (KKT) optimality conditions, we obtain the optimal values $\{\mathbf{w},b\}$, solution to the problem given by (1)–(3). Although F can be infinite dimensional (for instance, in the Gaussian kernel case), in which case $\mathbf{w}$ is also infinite dimensional, one of the KKT conditions tells us that $\mathbf{w}$ can actually be represented as a linear combination of projected datapoints $\boldsymbol{\phi}(\mathbf{x}_i)$, i.e. $\mathbf{w}$ lies in the span of $\{\boldsymbol{\phi}(\mathbf{x}_i),\ i=1,\ldots,P\}$: $\mathbf{w}=\sum_{i=1}^P y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)$. Additionally, as a result of the dual optimization, many of the Lagrange multipliers α_i usually become zero (sparse solution), the $\mathbf{x}_i$ corresponding to the nonzero ones are known as support vectors, which provides a more compact representation of $\mathbf{w}$ in terms of S Support Vectors ($S<P$): $\mathbf{w}=\sum_{i=1}^S y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)$. Also, this representation of $\mathbf{w}$ as a function of Support Vectors can be further simplified if we admit some error in the approximation, which is the principle of reduced set approaches described in [2] (or analogous methods [8,13,14]), which propose to obtain $\tilde{\mathbf{w}}$, an approximation to $\mathbf{w}$ using R ($R<S$) terms such that the error $\|\tilde{\mathbf{w}}-\mathbf{w}\|_2$ is acceptably small: $\tilde{\mathbf{w}}=\sum_{i=1}^R\beta_i\boldsymbol{\phi}(\mathbf{c}_i)$. Therefore, this set $\{\boldsymbol{\phi}(\mathbf{c}_i),\ i=1,\ldots,R\}$ of vectors in F that serves to approximate the solution $\mathbf{w}$, represents a reasonable base in F. We have named $\mathbf{c}_i$ patterns as centroids.

In our work we propose to select those centroids before training, by selecting an appropriate semiparametric machine, and not after training the full model, as in the case of the reduced set methods. Selecting them beforehand has the benefit of reduced training cost and allows us to deal with very reduced machines during the training process, as shown in [6,9,7]. Therefore, in what follows, we will assume that the hyperplane defined by $\mathbf{w}$ and b can be substituted by a linear combination of R projected patterns $\mathbf{c}_i$, such that the classifier becomes $o_j=sign(f(\mathbf{x}_j,\boldsymbol{\beta},b))$:

$$f(\mathbf{x}_j,\boldsymbol{\beta},b)=\sum_{i=1}^R\beta_i\boldsymbol{\phi}(\mathbf{c}_i)^T\boldsymbol{\phi}(\mathbf{x}_j)+b=\sum_{i=1}^R\beta_i k(\mathbf{c}_i,\mathbf{x}_j)+b=\boldsymbol{\beta}^T\mathbf{k}(\mathbf{x}_j)+b \tag{4}$$

where $\mathbf{k}(\mathbf{x}_j)=[k(\mathbf{c}_1,\mathbf{x}_j),k(\mathbf{c}_2,\mathbf{x}_j),\ldots,k(\mathbf{c}_R,\mathbf{x}_j)]^T$. Then (1) can be transformed into

$$L_{WLS}=\frac{1}{2}\boldsymbol{\beta}^T\mathbf{K}_c\boldsymbol{\beta}+\frac{1}{2}\sum_{i=1}^P a_ie_i^2 \tag{5}$$

where $e_i=y_i-f(\mathbf{x}_i,\boldsymbol{\beta},b)$, and a_i are weighting values to be computed as

$$a_i=\begin{cases}0, & e_iy_i<0\\ \dfrac{2C}{e_iy_i}, & e_iy_i\ge 0\end{cases} \tag{6}$$

Further detail about the derivation of expressions (5) and (6) can be found in [6,9], where it has also been shown that a two-step iterative procedure can be applied for minimizing (5) by firstly updating a_i values using (6), and secondly minimizing (5) by solving a system of linear equations of the form (weighted least squares problem[1]):

$$\begin{bmatrix}\mathbf{a}^T\mathbf{a} & \mathbf{a}^T\mathbf{K}\\ \mathbf{K}^T\mathbf{a} & \mathbf{K}^T\mathbf{D}_a\mathbf{K}+\mathbf{K}_c\end{bmatrix}\begin{bmatrix}b\\ \boldsymbol{\beta}\end{bmatrix}=\begin{bmatrix}\mathbf{a}^T\mathbf{y}\\ \mathbf{K}^T\mathbf{D}_a\mathbf{y}\end{bmatrix} \tag{7}$$

where $(\mathbf{K})_{i,j}=k(\mathbf{x}_i,\mathbf{c}_j)$, $(\mathbf{K}_c)_{i,j}=k(\mathbf{c}_i,\mathbf{c}_j)$, $\mathbf{y}=[y_1,\ldots,y_P]^T$, $\mathbf{a}=[a_1,\ldots,a_P]^T$ and $\mathbf{D}_a=diag\{a_i\}$, and repeating until convergence. For notational convenience, let us define the following intermediate variables:

$$\ddot{\mathbf{K}}=[\mathbf{1}|\mathbf{K}], \quad \ddot{\mathbf{K}}_c=\begin{bmatrix}0 & \mathbf{0}^T\\ \mathbf{0} & \mathbf{K}_c\end{bmatrix}, \quad \ddot{\boldsymbol{\beta}}=\begin{bmatrix}b\\ \boldsymbol{\beta}\end{bmatrix} \tag{8}$$

such that we can write (7) more compactly as

$$[\ddot{\mathbf{K}}^T\mathbf{D}_a\ddot{\mathbf{K}}+\ddot{\mathbf{K}}_c]\ddot{\boldsymbol{\beta}}=[\ddot{\mathbf{K}}^T\mathbf{D}_a\mathbf{y}] \tag{9}$$

We will name this iterated procedure for training the SSVM model as WLS-SVM from now on. Its convergence has been proved in [10,11].

3. The fast error estimation method

In the previous section we assumed that the set of centroids $\mathbf{c}_i$ is known before adjusting the SVM weights. In what follows we describe a procedure for selecting the best centroid to be added at every step during the semiparametric model construction. Let us assume that R centroids have already been incorporated to the model, and we need to identify the next one, $\mathbf{c}_{R+1}$. There is an optimal and brute-force (time consuming) greedy approach that consists in evaluating the cross-validation error that we obtain when every one of the potential candidates is added to the model. This direct approach is not applicable in practice due to the high computational cost, since the model weights would need to be computed too many times. We propose here to efficiently estimate such validation error for every new potential centroid, without the need of retraining every potential model. Let us assume that we are in the "R-th" growing step, we have stopped the WLS-SVC algorithm such that we are at the minimum of the cost function in (5) and optimal values for $\ddot{\boldsymbol{\beta}}$ and a_i are available. To evaluate the validation error when we add a new candidate $\mathbf{c}_{R+1}$ to the model, we need to solve the following set of equations:

$$\begin{bmatrix}\ddot{\mathbf{K}}^T\mathbf{D}_a\ddot{\mathbf{K}}+\ddot{\mathbf{K}}_c & \ddot{\mathbf{K}}^T\mathbf{D}_a\mathbf{k}_{R+1}+\mathbf{k}_c\\ \mathbf{k}_{R+1}^T\mathbf{D}_a\ddot{\mathbf{K}}+\mathbf{k}_c^T & \mathbf{k}_{R+1}^T\mathbf{D}_a\mathbf{k}_{R+1}+k_{cc}\end{bmatrix}\begin{bmatrix}\ddot{\boldsymbol{\beta}}'\\ \beta_{R+1}\end{bmatrix}=\begin{bmatrix}\ddot{\mathbf{K}}^T\mathbf{D}_a\mathbf{y}\\ \mathbf{k}_{R+1}^T\mathbf{D}_a\mathbf{y}\end{bmatrix} \tag{10}$$

where $(\mathbf{k}_{R+1})_i=k(\mathbf{x}_i,\mathbf{c}_{R+1})$, for $i=1,\ldots,P$, $(\mathbf{k}_c)_i=k(\mathbf{c}_i,\mathbf{c}_{R+1})$, for $i=1,\ldots,R$, and $k_{cc}=k(\mathbf{c}_{R+1},\mathbf{c}_{R+1})$. Strictly speaking, a_i values are also dependent on the new parameters $\ddot{\boldsymbol{\beta}}'$ and β_{R+1}, such that Eq. (10) would need to be repeatedly solved in the WLS-SVC way. This procedure would incur in an unacceptable computational

[1] When solving (9) with fixed weighting values $a_i=1$ (initial step), we obtain the solution known as least squares support vector machines (LS-SVM) [15], that, unless additional provision is taken, does not lead to sparse models.

load, so, if we assume that the present a_i values are a good approximation to the a_i values in the next iteration, then we can just simply solve (10) once and use the resulting model to estimate the validation error. We will validate this approximation in the experimental section. Furthermore, (10) can be solved as a function of the previously computed $\ddot{\boldsymbol{\beta}}'$ solution, obtained from WLS-SVC at the n-th stage, without any additional matrix inversion. For notational convenience, let us call $\mathbf{A} = \ddot{\mathbf{K}}^T\mathbf{D}_a\ddot{\mathbf{K}} + \ddot{\mathbf{K}}_c$, $\mathbf{b} = \mathbf{k}_{R+1}^T\mathbf{D}_a\ddot{\mathbf{K}} + \mathbf{k}_c^T$ and $c = \mathbf{k}_{R+1}^T\mathbf{D}_a\mathbf{k}_{R+1} + k_c$, such that we need to find the inverse matrix $\begin{bmatrix}\mathbf{P} & \mathbf{q}\\ \mathbf{q}^T & r\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{b}\\ \mathbf{b}^T & c\end{bmatrix}^{-1}$ or equivalently solve

$$\mathbf{A}\mathbf{q} + \mathbf{b}^T r = \mathbf{0} \tag{11}$$

$$\mathbf{b}^T\mathbf{q} + cr = 1 \tag{12}$$

$$\mathbf{A}\mathbf{P} + \mathbf{b}\mathbf{q}^T = \mathbf{I} \tag{13}$$

It is rather straightforward to operate (11) and (12) to obtain

$$r = 1/(c - \mathbf{b}^T\mathbf{z}) \tag{14}$$

$$\mathbf{q} = -\mathbf{z}r \tag{15}$$

where $\mathbf{z} = \mathbf{A}^{-1}\mathbf{b}$. Note here that $\mathbf{A}^{-1} = (\ddot{\mathbf{K}}^T\mathbf{D}_a\ddot{\mathbf{K}} + \ddot{\mathbf{K}}_c)^{-1}$ is available as a "by-product" of the WLS-SVC algorithm, since Eq. (9) is solved to obtain $\ddot{\boldsymbol{\beta}}$. Using these values in (13) we obtain the solution for $\mathbf{P}$:

$$\mathbf{P} = (\mathbf{I} - \mathbf{q}\mathbf{b}^T)\mathbf{A}^{-1} \tag{16}$$

If we use the equivalences (14)–(16) in Eq. (10) and after some simple algebraic workout, we finally obtain a compact solution for the weights of the new model as a function of the old ones:

$$\begin{bmatrix}\ddot{\boldsymbol{\beta}}'\\ \beta_{R+1}\end{bmatrix} = \begin{bmatrix}(\mathbf{I} - \mathbf{q}\mathbf{b}^T)\ddot{\boldsymbol{\beta}} + \mathbf{q}s\\ (s - \mathbf{b}^T\ddot{\boldsymbol{\beta}})r\end{bmatrix} \tag{17}$$

where $s = \mathbf{k}_{R+1}^T\mathbf{D}_a\mathbf{y}$. Given the new model weights in (17) it is straightforward to evaluate the classification error in the validation set and, therefore, select the candidate centroid that produces the lowest error. The procedure of training weights using the WLS-SVC approach and selecting the best centroid to be added using the FEE method is continued until a stopping criterion is

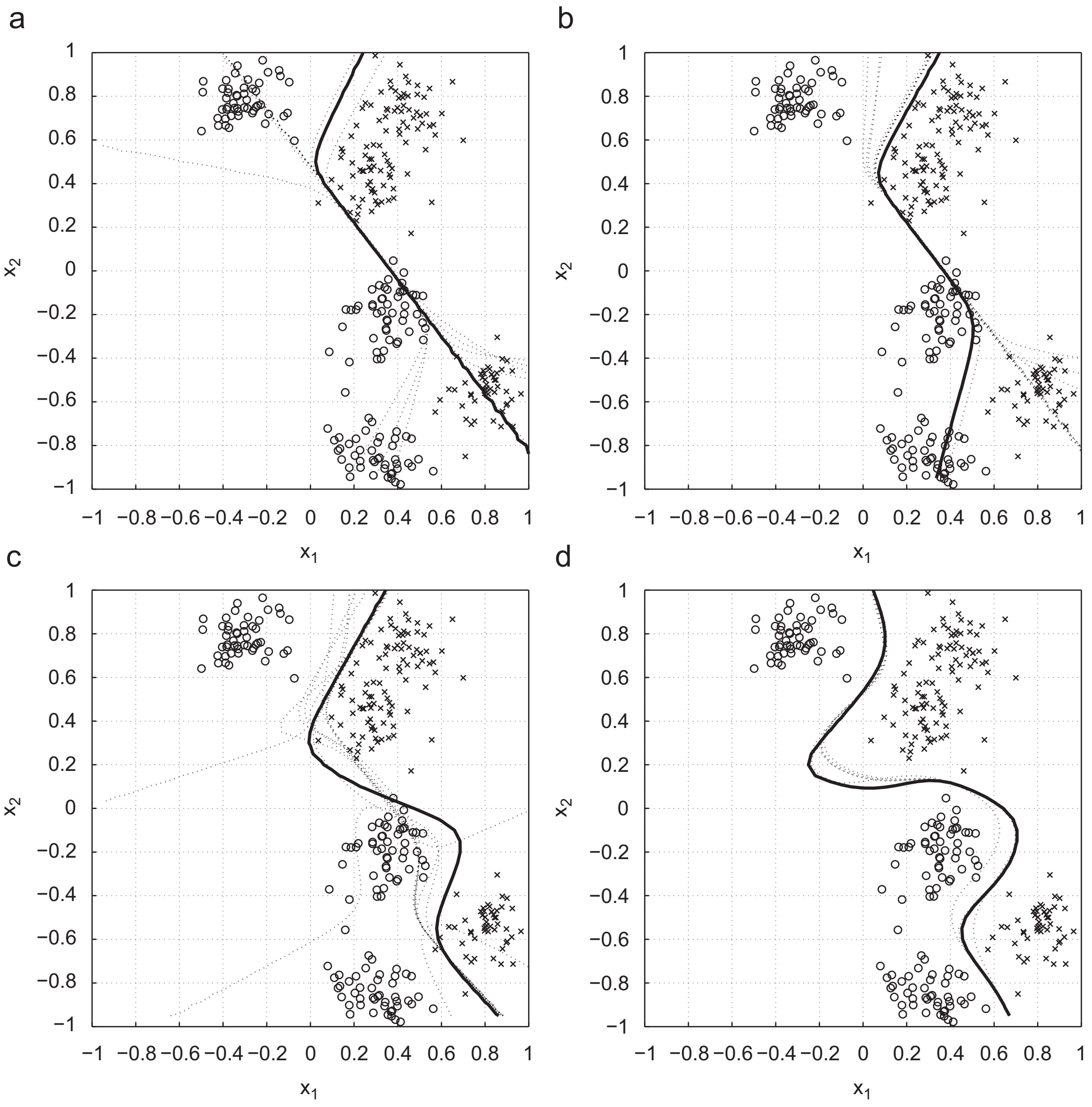


Fig. 1. Decision boundaries estimated using the FEE procedure at different steps of the algorithm ($n = 3$ (a), $n = 5$ (b), $n = 7$ (c), $n = 10$ (d)) are shown as dotted lines, whereas the actual boundary selected by the algorithm at every stage is marked with a thicker plain line.

Table 1
The fast error estimation support vector machine (FEE-SVM) algorithm.

- **Step 0**: Initialize the model by choosing as the two first centroids ($R=2$) the mean value of the training patterns belonging to every class.
- **Step 1**: Compute weights $\ddot{\boldsymbol{\beta}}$ using the WLS-SVC algorithm:
  - *Step* 1.0: Initialize $a_i=1$, $\forall i=1,\dots,P$
  - *Step* 1.1: Build the following matrices and vectors:

$$\mathbf{K}:(\mathbf{K})_{i,j}=k(\mathbf{x}_i,\mathbf{c}_j);\ i=1,\dots,P;\ j=1,\dots,R$$
$$\mathbf{K}_c:(\mathbf{K}_c)_{i,j}=k(\mathbf{c}_i,\mathbf{c}_j);\ i=1,\dots,R;\ j=1,\dots,R$$
$$\mathbf{D}_a:(\mathbf{D}_a)_{i,i}=a_i;\ i=1,\dots,P$$
$$\mathbf{y}=[y_1,\dots,y_P]^T$$
$$\mathbf{1}=[1,\dots,1]^T$$
$$\ddot{\mathbf{K}}=[\mathbf{1}|\mathbf{K}]$$
$$\ddot{\mathbf{K}}_c=\begin{bmatrix}0 & \mathbf{0}^T\\ \mathbf{0} & \mathbf{K}_c\end{bmatrix}$$

  - *Step* 1.2: Obtain optimal weights $\ddot{\boldsymbol{\beta}}=[\ddot{\mathbf{K}}^T\mathbf{D}_a\ddot{\mathbf{K}}+\ddot{\mathbf{K}}_c]^{-1}[\ddot{\mathbf{K}}^T\mathbf{D}_a\mathbf{y}]$
  - *Step* 1.3: Compute outputs $o(\mathbf{x}_i)=\sum_{r=1}^{R}\beta_i k(\mathbf{x}_i,\mathbf{c}_r)+b$ and errors $e_i=y_i\text{-}o(\mathbf{x}_i)$
  - *Step* 1.4: Update weighting values:

$$(a_i=\begin{cases}0; & e_iy_i<0\\ M; & 0\le e_iy_i\le C/M;\quad M=10^9\\ \dfrac{C}{e_iy_i}; & e_iy_i>C/M\end{cases}$$

  - *Step* 1.5: Evaluate the WLS-SVC stopping condition $\|\ddot{\boldsymbol{\beta}}^{(k+1)}\text{-}\ddot{\boldsymbol{\beta}}^{(k)}\|_2<10^{-5}$. If convergence is achieved, then proceed to **Step 2**, otherwise go back to *Step* 1.1.

- **Step 2**: Stop growing if the validation error does not decrease 20 steps after the minimum, otherwise identify the new centroid to be added to the base:
  - *Step* 2.1: Randomly select 10 candidate centroids among the support vectors.
  - *Step* 2.2: For every candidate $\mathbf{c}_{R+1}$ estimate the new model weights:

$$\mathbf{k}_{R+1}:(\mathbf{k}_{R+1})_i=k(\mathbf{x}_i,\mathbf{c}_{R+1});\ i=1,\dots,P;$$
$$\mathbf{k}_c:(\mathbf{k}_c)_i=k(\mathbf{c}_i,\mathbf{c}_{R+1});\ i=1,\dots,R;$$
$$k_{cc}=k(\mathbf{c}_{R+1},\mathbf{c}_{R+1})$$
$$\mathbf{A}^{-1}=(\ddot{\mathbf{K}}^T\mathbf{D}_a\ddot{\mathbf{K}}+\ddot{\mathbf{K}}_c)^{-1}(\text{available from previous WLS}-\text{SVC step})$$
$$\mathbf{b}=\mathbf{k}_{R+1}^T\mathbf{D}_a\ddot{\mathbf{K}}+\mathbf{k}_c^T$$
$$c=\mathbf{k}_{R+1}^T\mathbf{D}_a\mathbf{k}_{R+1}+k_c$$
$$\mathbf{z}=\mathbf{A}^{-1}\mathbf{b}$$
$$r=1/(c\text{-}\mathbf{b}^T\mathbf{z})$$
$$\mathbf{q}=\text{-}\mathbf{z}r$$
$$s=\mathbf{k}_{R+1}^T\mathbf{D}_a\mathbf{y}\begin{bmatrix}\ddot{\boldsymbol{\beta}}'\\ \beta_{R+1}\end{bmatrix}=\Big[(\mathbf{I}\text{-}\mathbf{q}\mathbf{b}^T)\ddot{\boldsymbol{\beta}}+\mathbf{q}s(s\text{-}\mathbf{b}^T\ddot{\boldsymbol{\beta}})r\Big]$$

  - *Step* 2.3: Evaluate the validation error of every model, and select as $\mathbf{c}_{R+1}$ the one with the lowest classification error on the validation set.
  - *Step* 2.4: Go back to **Step 1**.

met. In what follows we have used a simple rule: stop adding nodes if the minimum on the training error does decrease after 20 steps. Such minimum value on the validation set and the corresponding machine size are stored to be used i n the N-fold cross-validation procedure for hyper-parameter selection.

As a graphical illustration of the FEE-SVM growing procedure, we have depicted in Fig. 1 several intermediate steps observed when solving a synthetic two-dimensional binary classification problem. We represent patterns belonging to either class $+1$ or -1 as "$\circ$" or "$\times$" symbols. At a given stage, every one of the 10 candidate centroids, when added to the present model, produces a different classifier, which can be efficiently estimated using Eq. (17). We have depicted the decision boundary of every such estimated classifier using dotted lines. Among the evaluated models, the decision boundary of the winning model at every stage is drawn using a thicker continuous line. Note how, step by step, the procedure is able to build a boundary aiming at minimizing the global classification error. We have summarized in Table 1 the proposed FEE-SVM algorithm and in the next section we will evaluate its performance.

4. Experiments

We have benchmarked FEE-SVM against SGMA-SVM and the standard SVM (trained using the LibSVM software[4]) on several datasets from the UCI machine learning repository. Hyperparameters have been selected using fivefold cross-validation for all algorithms using a grid in C and σ (a Gaussian kernel has been used in the experiments). We have evaluated values

Table 2
Performance of LibSVM, SGMA-SVM and FEE-SVM on the UCI datasets.

Dataset	N_{train}	N_{var}	Classification error (%)		
			LibSVM	SGMA-SVM	FEE-SVM
Heart	202	13	13.8	13.3	13.7
Ripley	250	2	9.7	10.1	10.4
Breast	466	9	4.3	4.3	4.6
Kwok	500	2	12.1	12.3	12.2
Pima	512	8	21.9	23.9	23.2
Image	1540	18	1.9	2.9	2.6
Add	1887	1558	3.8	3.3	3.3
Spam	3680	57	5.9	6.6	6.9
Handdigit	3823	64	1.1	1.8	2.0
Waveform	4000	21	8.3	8.5	8.4
Landsat	4435	36	0.8	0.82	0.64
Twonorm	5920	20	2.8	2.7	2.7
Ringnorm	5920	20	1.6	2.1	2.4
Adult	30 162	33	24.6	24.2	24.1
Average	4521.2	132.9	8.04	8.34	8.36

The number of training patterns (N_{train}), number of input variables (N_{var}) and classification error (CE) are shown. Note how the compact solutions obtained with SGMA-SVM and FEE-SVM provide comparable performance (on average 8.34 vs. 8.36), only slightly worse than the full SVM (8.04 on average) obtained with LibSVM.

Table 3
Performance of LibSVM, SGMA-SVM and FEE-SVM on the UCI datasets.

Dataset	Machine size			Training time (s)		
	LibSVM	SGMA	FEE	LibSVM	SGMA	FEE
Heart	106	**15**	17	26	1695	**719**
Ripley	77	24	**16**	335	730	**397**
Breast	33	19	**15**	39	874	**757**
Kwok	247	**23**	24	144	791	**487**
Pima	252	20	**18**	280	3628	**2264**
Image	215	107	**71**	709	10 833	**8972**
Add	460	45	**34**	49 766	98 312	**12 805**
Spam	613	71	**67**	12 132	**16 725**	28 969
Handdigit	676	84	**73**	15 053	209 056	**30 555**
Waveform	1218	54	**41**	7227	60 795	**32 348**
Landsat	204	50	**46**	4713	16 811	**10 403**
Twonorm	624	**20**	23	12 690	76 802	**31 092**
Ringnorm	779	102	**99**	12 341	202 261	**57 968**
Adult	15 016	**13**	16	1 948 750	**173 520**	208 073
Average	1466	46	**40**	147 443	62 345	**30 415**

The resulting machine size and training time in seconds (1000 runs) are shown. The LibSVM results were obtained with a C compiled code, whereas SGMA-SVM and FEE-SVM have been implemented using Matlab scripts, therefore the LibSVM training times are not directly comparable. We focus here on the comparison between both compact methods, SGMA-SVM and FEE-SVM. Note how FEE-SVM provides machines smaller than SGMA-SVM, on average, 13% smaller (40 vs. 46). Additionally, the training times are also favorable to the FEE-SVM method. Finally, note the poor scalability of LibSVM when the dataset is large (Adult).

$\sigma = 2^n\sqrt{N}, n = -6$,- $5,\ldots,2,3$, (where N is the input dimension), and $C = 10^n, n = -4$,- $3,\ldots,5$, accounting for 100 (C,σ) pairs in total. Since we have used a fivefold approach, a total of 500 training executions have been carried out for every dataset in a first round. After finding the (C, σ) pair that provides the minimal cross-validation error, a second finer grid in the (C,σ) space around the minimum is evaluated, using another 500 runs. Therefore, to obtain the optimal hyperparameters for every dataset, a total of 1000 training executions have been run.

In Table 2 we provide more details about the datasets used. The number of training patterns ranges from 202 to 30 162 and the number of input variables from 2 to 1558. Also in Table 2 we include the classification error (CE) rates in the test set (obtained by averaging over 100 different trials). We observe that both SGMA-SVM and FEE-SVM provide comparable performance (on average 8.34 vs. 8.36), only slightly worse than the full SVM model (8.04 on average). This slight reduction in performance is the price to be paid for the compactness of the reduced models.

In Table 3 we present results concerning final machine size and computational cost. First of all, note the size reduction with respect to the full SVM method, the models resulting from SGMA-SVM and FEE-SGMA are 1–2 orders of magnitude smaller (3 orders in the Adult case). The model size obtained with the proposed FEE-SVM method is 13% smaller on average (40 vs. 46) with respect to the SGMA-SVM approach, which reinforces our hypothesis that estimating the validation errors using the FEE approach is a better approach than simply modelling the input data space, as in the SGMA-SVM case. Concerning computational cost, LibSVM is the most efficient method, but it is a compiled C program, whereas the other two algorithms have been implemented using Matlab scripts, which are slower. Nevertheless, in the Adult case, the computation time is much larger in the LibSVM case than in the other methods, which illustrates another benefit of semiparametric models trained with the WLS-SVC algorithm: they provide better scalability than quadratic programming training methods when the number of training patterns is large. The comparison between SGMA-SVM and FEE-SVM in computational cost also favors the latter.

5. Conclusions and further work

We have proposed a method for selecting the centroids to be added when growing a semiparametric SVM model, to obtain compact SVM models at a very competitive computational cost. The proposed FEE-SVM method proposes to select the best centroid by estimating the validation error. It has proved to obtain smaller models than previously proposed approaches, where an optimal input space data representation criterion was used (sparse greedy matrix approximation, SGMA-SVM). The FEE-SVM benefits from a very efficient inverse computation at step "$n+1$" , given that the inverse at stage "n" is available as a by-product of the algorithm for updating the weights (WLS-SVC) at that stage, also representing a computational cost reduction with respect to the SGMA-SVM approach. As a further work, we propose to extend this method to other kernel models such as Gaussian processes or kernel partial least squares, that could also benefit from an improved centroid selection at every stage, to obtain maximally compact resulting models.

References

[1] G.H. Bakir, J. Weston, B. Schölkopf, Learning to find pre-images, Advances in Neural Information Processing Systems 16 (7) (2004) 449–456.
[2] C.J.C. Burges, B. Schölkopf, Improving the accuracy and speed of support vector learning machines, in: M. Mozer, M. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems, vol. 9, MIT Press, Cambridge, CA, 1997, pp. 375–381.
[3] J.T. Kwok, I.W. Tsang, The pre-image problem in kernel methods, IEEE Transactions on Neural Networks 15 (6) (2004) 1517–1525.
[4] ⟨http://www.csie.ntu.edu.tw/cjlin/libsvm/⟩.
[6] A. Navia-Vázquez, F. Pérez-Cruz, A. Artés-Rodríguez, A. Figueiras-Vidal, Weighted least squares training of support vector classifiers leading to compact and adaptive schemes, IEEE Transactions on Neural Networks 12 (5) (2001) 1047–1059.

[7] A. Navia-Vázquez, Compact multiclass support vector machines, Neurocomputing 71 (1–3) (2007) 400–405.
[8] E. Osuna, F. Girosi, Reducing the run-time complexity in support vector regression, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods Support Vector Learning, MIT Press, Cambridge, MA, 1999, pp. 271–284.
[9] E. Parrado-Hernández, J. Arenas-García, I. Mora-Jiménez, A.R. Figueiras-Vidal, A. Navia-Vázquez, Growing support vector classifiers with controlled complexity, Pattern Recognition 36 (7) (2003) 1479–1488.
[10] F. Pérez-Cruz, Máquina de Vectores Soporte Adaptativa y Compacta, Ph.D. Thesis, Universidad Politécnica de Madrid, 2000 (in Spanish).
[11] F. Pérez-Cruz, C. Bousoño-Calzón, A. Artés-Rodríguez, Convergence of the IRWLS procedure to the support vector machine solution, Neural Computation 17 (2005) 7–18.
[12] B. Schölkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[13] B. Schölkopf, P. Knirsch, A. Smola, C. Burges, Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces, Neural Computation 10 (1998) 1299–1319.
[14] A.J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: P. Langley (Ed.), Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000, pp. 911–918.
[15] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific Publishing Co., Singapore, 2002.
[16] B. Tang, D. Mazzoni, Multiclass reduced-set support vector machines, in: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 2006.
[17] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

Angel Navia-Vázquez received his degree in Telecommunications Engineering in 1992 (Universidad de Vigo, Spain), and finished his PhD, also in Telecommunications Engineering in 1997 (Universidad Politécnica de Madrid, Spain). Since 2001 he is an Associate Professor at the Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Spain. His research interests are focused on new architectures and algorithms for nonlinear processing as well as their application to content processing and management, communications and data mining. He has (co)authored 19 international refereed journal papers in these areas, several book chapters, more than 50 conference communications, and participated in more than 20 research projects. He is IEEE (Senior) Member since 1999 and Associate Editor of IEEE Transactions on Neural Networks (2004–2009).

Roberto Díaz Morales received his Telecommunications Engineering degree from the University Carlos III of Madrid (Spain) in 2006. Until 2008 he worked in Sun Microsystems in the web services area. Currently he is working on his PhD and his interests are focused on machine learning.