Growing Support Vector Classifiers with Controlled Complexity ⋆

E. Parrado-Hernández *, I. Mora-Jiménez, J. Arenas-García, A.R. Figueiras-Vidal, A. Navia-Vázquez

Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, Avda de la Universidad 30, 28911 Leganés-Madrid, Spain

Abstract

Semiparametric Support Vector Machines have shown to present advantages with respect to nonparametric approaches, in the sense that generalization capability is further improved and the size of the machines is always under control. We propose here an incremental procedure for Growing Support Vector Classifiers, which serves to avoid an *a priori* architecture estimation or the application of a pruning mechanism after SVM training. The proposed growing approach also opens up new possibilities for dealing with multi-kernel machines, automatic selection of hyperparameters, and fast classification methods. The performance of the proposed algorithm and its extensions is evaluated using several benchmark problems.

Key words: Support Vector Classifiers, incremental, compact, multikernel, controlled size, Support Vector Machines (SVM)

⋆ This work has been partially supported by Spain CICYT grant TIC 1999-0216.

* Corresponding author. Fax: +34 916248749
Email address: `emipar@tsc.uc3m.es` (E. Parrado-Hernández).

1 Introduction

Support Vector Classifiers (SVCs) find maximal margin boundaries between classes, minimizing the structural risk [1] [2], and therefore endowing the classifier with excellent generalization capabilities. Moreover, SVCs automatically determine the machine architecture in terms of a subset of the input patterns (Support Vectors (SVs)). SVCs have proven to successfully solve many real world applications, such as text classification [3], image recognition [4] and bioinformatics [5], among many others.

However, SVCs present certain drawbacks in comparison with other machine learning techniques that may reduce their usability in some domains. First, the classical training methods —relying on Quadratic Programming (QP) techniques— have shown to be computationally costly [6]. In addition to this, the classifier size usually results very high (sometimes comparable to a significant fraction of the amount of data). Furthermore, the training proceeds in a closed "black-box" form, and the classifier architecture is obtained after applying a deterministic optimization procedure on the training set. This difficults the development of modified, more flexible, versions of the algorithm. In this sense, it seems vital being able to monitor/control performance and complexity during training, such that new criteria can be gracelly imposed. Our incremental formulation will provide such flexibility.

Several approaches to overcome these inconveniences have appeared in the literature. With respect to the compactness of the machines, in [7] an already QP-trained SVC is approximated by a smaller one minimizing the approximation error. Another strategy consists on simplifying the classification boundary with a Support Vector Regressor [8]. Although both alternatives yield simpler machines, the computational cost is significantly increased since they first solve a complete SVM problem and then postprocess the architecture so as to approximate it with fewer nodes. On the other hand, some techniques such as *chunking* or *shrinking* [9] can help to reduce the training cost; those alternatives still retain the “black-box” characteristic, and finally lead to the QP solution.

Suboptimal constructive methods have also been proposed to reduce the complexity of the machine while controlling the training cost. These algorithms are based on the Sparse Greedy Approximation (SGA) [10], and they allow to control the complexity of the QP problem to be solved. Nevertheless, these constructive algorithms try to achieve a compact representation of the input data in the projection space, like those of Kernel PCA [11], and then they solve a classical SVC. Thus, the size of the resulting architecture depends on the complexity of the representation of the dataset, and this does not necessarily have to be related to the complexity of the classification problem to be solved [12].

Alternative algorithms relying on Weighted Least Squares (WLS) instead of QP have already succeeded in reducing training cost: IRWLS [6] and WLS-SVC [13], the latter providing compact (semiparametric) solutions. However, WLS-SVC needs to be fed with an *a priori* estimation of the machine architecture, solved in [13] using nonlinear PCA or clustering for the Gaussian kernel case, or with one of the constructive methods based on SGA.

As an alternative to this —probably suboptimal— two-stage approach (architecture estimation plus training), we propose here a growing procedure for

SVCs —GSVC algorithm— such that the trade-off between performance and machine size is always under control, directly depending on the complexity of the addressed problem. The iterative nature of GSVC allows the creation of implicit Reduced-Set-like fast classification methods at no extra cost.

The problem of selecting kernel hyperparameters is still an open issue under the GSVC scheme, since, for instance in the Gaussian kernel case, optimal value σ has to be found before training using an off-line cross-validation procedure. GSVC facilitates the introduction of a multikernel approach, MK-GSVC. For the Gaussian kernels, the proposed MK-GSVC method starts with initial large spreading value for σ (which provide initial machines working at a coarse level of detail), and progressively decreases this value in newly added kernels, such that finer solutions are obtained as the machine grows (multiresolution approach).

The rest of the paper is organized as follows: Section 2 briefly reviews SVM and WLS-SVC formulation. Section 3 is devoted to the presentation of the growing (GSVC) and multikernel (MK-GSVC) algorithms, and the fast classification method (FC-GSVC). Experimental results concerning the proposed algorithms in comparison with an implementation of the classical SVC are offered in Section 4. Finally, Section 5 collects the main conclusions and identifies lines for future research.

2 SVMs and WLS-SVC training

Let us assume we have to solve a binary classification problem described by labeled training pairs consisting of a pattern and a target $(\{\vec{x}_i, y_i\}, i = 1, \ldots, N)$, with $y_i \in \{-1, +1\}$.

SVM projects input patterns $\vec{x}_i$ with a nonlinear function $\vec{\phi} : \vec{x} \to \vec{\phi}(\vec{x})$ onto a higher dimension space $\mathcal{F}$ and, then, it separates the data in $\mathcal{F}$ $(\{\vec{\phi}(\vec{x}_i)\})$ with a maximal margin hyperplane. Therefore, the classifier is given by

$$f(\vec{x}) = \mathrm{sign}(\vec{w}^T \vec{\phi}(\vec{x}) + b) \tag{1}$$

and parameters $\vec{w}$ and b are obtained through the minimization of functional L_p in (2), that maximizes the margin of the classification boundary [1]:

$$L_p \equiv \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i y_i(\vec{w}^T\vec{\phi}(\vec{x}_i) + b) - \sum_{i=1}^{N}\eta_i\xi_i \tag{2}$$

$$\text{under } \alpha_i, \eta_i, \xi_i \geq 0$$

where $\{\alpha_i\}$ are Lagrange multipliers and $\{\xi_i\}$ are a set of slack variables that are introduced to solve non-separable cases [1].

Unfortunately, in many interesting cases, either $\vec{\phi}(\cdot)$ is unknown or the dimension of $\mathcal{F}$ is infinite, what makes $\vec{\phi}(\cdot)$ impossible to calculate, so that $\vec{w}$ cannot be directly obtained. The usual procedure in such cases is to solve the problem using its alternative dual formulation. Since the solution of the linear classifier in $\mathcal{F}$ only involves inner products of vectors $\vec{\phi}(\vec{x}_i)$, we can always use the *kernel trick* [14], which consists on expressing the inner product in $\mathcal{F}$ as an evaluation of a kernel function in the input space:

$$< \vec{\phi}(\vec{x}), \vec{\phi}(\vec{y}) >= k(\vec{x}, \vec{y})$$

This way, we do not need to explicitely know $\vec{\phi}(\cdot)$ but just its associated kernel $k(\vec{x}, \vec{y})$. Many kernels are used in practice [15], but in this paper we will focus on the Gaussian

$$k(\vec{x}, \vec{y}) = \exp\left(-\frac{\|\vec{x} - \vec{y}\|^2}{2\sigma^2}\right)$$

When expressed in terms of kernels, the classifier results

$$f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i k(\vec{x}_i, \vec{x}) + b\right) \tag{3}$$

If coefficients $\{\alpha_i\}$ are obtained after a QP optimization of functional (2), we have the QP-SVM methods [1].

According to (3), those $\vec{x}_i$ with $\alpha_i > 0$ are called Support Vectors (SVs), and the size of the machine is equal to the number of SVs (N_{SV}). In cases where classes highly overlap many SVs are found, what results in very large machines.

Alternative algorithms, exploiting the fact that SVs usually span a subspace of dimension $R << N_{SV}$, can accurately approach the SVM solution with a compact machine. In this sense, WLS-SVC builds a semiparametric approximation to the SVC from an *a priori* estimated architecture of R nodes [13]. The formulation of the WLS-SVC is as follows: let us assume a parametric solution to the SVM classification boundary of the form

$$\hat{f}(\vec{x}) = \text{sign}\left(\sum_{k=1}^{R} \beta_k \vec{\phi}(\vec{c}_k)\vec{\phi}(\vec{x}) + b\right) = \text{sign}\left(\sum_{k=1}^{R} \beta_k k(\vec{c}_k, \vec{x}) + b\right) \tag{4}$$

where $\{\vec{c}_k\}$ are R centroids corresponding to the principal directions of the subspace of $\mathcal{F}$ spanned by the training patterns, selected by means of any clustering algorithm. Coefficients $\{\beta_k\}$ are randomly initialized and solved by an iterative application of WLS [13] until convergence. Basically, WLS-SVC operates with a matrix K of kernels of size $N \times R$, with $K_{ij} = k(\vec{x}_i, \vec{c}_j)$, and minimizes (2) with respect to $\{\beta_k\}$ for the whole training set. Simultaneously, it identifies the SVs and satisfies the constraints of the SVM optimization problem so that the classification boundary asymptotically approaches that found by the original SVM as R becomes high enough. Thus, from now on, WLS-SVC is going to be considered as an algorithm that is fed with matrix K and the labels of the training set, $\{y_i\}$, and outputs the parametric machine, $\{\beta_k\}$, b, and coefficients $\{\alpha_i\}$ of the SVM. In [13] the WLS-SVC algorithm is explained in more detail and some experimental results are provided, showing its advantages over the classical methods to design a SVC. Among these advantages, it is worth remarking its computational and memory efficiency (WLS-SVC involves inversion of matrices $R \times R$ and storage of matrices $N \times R$ with $R << N_{SV} \leq N$, while QP-SVM works with matrices $N \times N$ and $N_{SV} \times N_{SV}$). Unfortunately, estimating R centroids *a priori* is suboptimal, since no information about where the SVs are is available. But our formulation enables the development of growing procedures which represent a good trade-off between initial uncertainity about the problem and compactness of the final machine.

3 Growing Support Vector Classifiers

Our proposal to construct the GSVC consists on starting with a reduced machine (with few centroids randomly selected) and, at each step of the algorithm, adding new ones to improve the classification of the (so far) incorrectly classified patterns. Every time the architecture is grown, WLS-SVC algorithm allows to update the solution, without needing to solve from scratch.

The growth can be stopped according to a criterion imposed either on the size or on the performance of the machine. This allows to control the trade-off between complexity of the machine and accuracy.

3.1 Growing Support Vector Classifier Algorithm: GSVC

For initialization of the GSVC algorithm, the first M nodes [1] (tipically $M = 6, 8$) are picked up at random from the training set, $\frac{M}{2}$ from every class, since no information about the boundary is available yet. Small values of M involve a more gradual growing of the classifier, with more growing iterations, while larger values of M result in less growing steps although the final machine size may be larger due to the introduction of some suboptimal nodes. Then, an initial kernel matrix K_0 is computed and the initial hyperplane $\vec{\beta}(0)$ and coefficients $\vec{\alpha}(0)$ are obtained via WLS-SVC.

The preliminary classification error can be used to select new centroids to increase the representational capabilities of the machine, by incorporating them into the architecture and updating weights. Good procedures exist to find (before training) a near-optimal subset of representational axis $\{\vec{\phi}_k\}$ in $\mathcal{F}$ for a given dataset, for instance by means of nonlinear PCA [16], such that they define an orthonormal basis. Unfortunately, this is difficult here: first, we need to find the Singular Value Decomposition (SVD) of a $N \times N$ matrix. Second, we need to find elements $\{\vec{c}_k\}$ in the input space such that $\vec{\phi}(\vec{c}_k) \simeq \vec{\phi}_k$ to obtain a final machine of form (4). Although this procedure has already been formulated as an Expectation-Maximization (EM) algorithm [16], its correct

[1] Since we will restrict our formulation to Gaussian kernels, and the classifier in (4) can be interpreted as a single layer Neural Network, we will use both "centroid" and "node" to refer to every new $k(\vec{x}, \vec{c}_i)$ term.

application is not a trivial task. When this problem is stated as a clustering in input space ([16], [13]), it requires to set the number of centroids R 'a priori', and to solve a large clustering problem; perhaps wasting many centroids to model regions with little interest (far from the decision boundary). To overcome this, we will follow here the principles of Boosting [17] to select the new $\{\vec{\phi}(\vec{c}_k)\}$. For a given architecture (step "n"), we force the training procedure to concentrate on missclassified or close to the boundary (inside the margin) patterns, thereby improving the representation in $\mathcal{F}$ of this region of the input space. SVC training (also in its WLS-SVC implementation) provides a list of those patterns that are critical for training (SVs): patterns with $\alpha_i > 0$. Hopefully, by concentrating on these hard-to-learn patterns, the overall performance of the machine can be improved, since they represent a sufficient set of data for solving the problem at hand. Nevertheless, we know that, in problems with highly overlapped classes, a large number of SVs can be found in the overlapping areas. Therefore, an additional criterion has to be used to avoid selecting a lot of patterns in the same region. We propose the following combined criterion: among those patterns with $\alpha_i > 0$, choose randomly between the ones with smaller maximal projection onto the actual basis, what enforces maximal orthogonality with respect to the existing axes. This quantity can be easily estimated using the kernel matrix, by simply finding the maximum of the kernel values associated to every pattern, since every kernel itself represents the projection onto one axis. According to [6], not always the "best" candidates for becoming a centroid following any given criterion turn out to be the most suitable once incorporated into the machine. In consequence, GSVC designs a pool of candidates to become centroid from the hard-to-learn patterns, and then it selects the next centroids randomly from the pool. The growth is continued until a convergence criterion is reached.

Note that, every time the classifier is expanded with new nodes, the GSVC algorithm only updates the parameters for step "n" using those at step "$n-1$", such that it does not need re-training from scratch. This would be hard to implement under a QP formulation. Furthermore, all the kernels computed at step "n" do not need to be re-computed at following steps, because their centroids are kept. In addition, it has to be pointed out that, although the procedure for selecting the new centroids is a greedy one (at each step pick up the centroids from the best candidates), the solution that provides the Structural Risk Minimization (SRM) is achieved at each iteration of the algorithm for the given centroids. This is guaranteed by WLS-SVC determining an asymptotic approach to the SVC from the current centroids of the machine. So, WLS-SVC updates globally all the $\{\beta_k\}$ and solve SRM for the architecture at hand. In [13], it is shown that sensitivity with respect to the centroids is very low, ensuring the quality of the current parametric approximation.

To avoid overfitting and to get good generalization characteristics, we stop the algorithm by means of a validation procedure that detects when performance stops increasing. The validation is applied on a reserved subset (about 20%) of the training data that is not fed into WLS-SVC until the architecture is completely determined. Once the growth has finished, the training and validation subsets are merged and WLS-SVC is invoked for a last update of the classifier weights. Proceeding like this, WLS-SVC guarantees that the solution achieved by GSVC is a good approximation to that of SVC, and often this validation stopping criterion serves to excell classic SVC generalization capability. In order to maintain a minimum degree of performance for the growing algorithm, stopping by cross-validation is only applied when the number of SVs found by WLS-SVC after several growing steps is observed to stabilize, i.e., the GSVC is close enough to the SVC solution. As stated before, this criterion on the classification error on the validation set can be combined with additional constraints on the size of the machine.

Computational cost is saved by the application of *shrinking* of the training set [9]. This technique consists on identifying those training patterns that, iteration after iteration, do not become SVs and temporarily removing them from the training set. Kernel matrix is merely updated and not recalculated at all. Anyway, those temporarily removed patterns are monitored after each iteration and returned to the "active" training set if they become SVs.

We summarize the GSVC algorithm in Table 1.

3.2 Mixture of kernels to solve a multi-resolution classifier: MK-GSVC

In some applications, it is not clear —*a priori*— which kernel function is the most appropiate, and it might be desirable to train a more flexible SVM by combining different kernels to solve a given problem. In what follows we will restrict to the Gaussian case, where combining kernels with different σ value can be interpreted as a multiresolution approach, observing the problem at different scales of detail. Such incremental multikernel scheme could start at a coarse resolution and try to solve the problem by adding new nodes with the same value of σ until the classifier saturates (according to the evolution of the Classification Error (CE) on the validation set) and then reduce σ, and continue adding nodes. This is repeated until no improvement in the CE of the validation set is obtained. The general effect is a fine tuning of the machine on the more difficult-to-learn regions of the classification boundary. This way, the previous effort of pre-estimating a suitable value of parameter σ that ensures a good performance of the classifier is skipped.

It is necessary to reformulate the algorithms to support multi-kernel, by combining several discriminant functions with different kernels and globally maximizing the margin of the resulting machine. The multi-kernel SVC is:

$$f(\vec{x}) = \operatorname{sign}\left(\sum_{j=1}^{J} \vec{w}_j^T \vec{\phi}_j(\vec{x}) + b\right) \tag{5}$$

where $\vec{\phi}_j(\cdot)$ are the map functions corresponding to the J different kernels and b includes all the bias terms. Coefficients $\vec{w}_j^T$ are obtained by jointly maximizing the margins of the classifier in each of the spaces $\mathcal{F}_j$ ($\mathcal{F}_j$ corresponds to projection $\vec{\phi}_j(\cdot)$), as usual in SRM:

$$\begin{aligned} L_{MK} = & \sum_{j=1}^{J} \frac{\delta_j}{2} \|\vec{w}_j\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\sum_{j=1}^{J} \vec{w}_j^T \vec{\phi}_j(\vec{x}_i) + b \right) - 1 + \xi_i \right] \\ & - \sum_{i=1}^{N} \eta_i \xi_i + \sum_{i=1}^{N} \xi_i \end{aligned} \tag{6}$$

where coefficients $\{\delta_j\}$ represent a set of (possible) weights for the combination of kernels. Then, using

$$e_i = y_i - \left(\sum_{j=1}^{J} \vec{w}_j^T \vec{\phi}_j(\vec{x}_i) + b\right) \tag{7}$$

$$a_i = \frac{2\alpha_i y_i}{e_i} \tag{8}$$

and minimizing (6) we arrive to

$$\delta_j \vec{w}_j - \Phi_j^T D_a \left[\vec{y} - \Phi W - \vec{1} b\right] = \vec{0} \qquad j = 1, \ldots, J \tag{9}$$

$$-\vec{a}^T \left[\vec{y} - \Phi W - \vec{1} b\right] = 0 \tag{10}$$

where $\Phi_j = \left[\vec{\phi}_j(\vec{x}_1) \cdots \vec{\phi}_j(\vec{x}_N)\right]^T$, D_a is a diagonal matrix with $(D_a)_{ii} = a_i$,

$\Phi = [\Phi_1 \cdots \Phi_J]$, $W = \left[\vec{w}_1^T \cdots \vec{w}_J^T\right]^T$, $\vec{1} = [1 \cdots 1]^T$, and $\vec{a} = [a_1 \cdots a_N]^T$. We can further combine the equations in (9) obtaining:

$$\Lambda W - \Phi^T D_a \left[\vec{y} - \Phi W - \vec{1} b\right] = \vec{0} \tag{11}$$

where Λ is a diagonal matrix defined as

$$\Lambda = \begin{bmatrix} A_1 I_1 & 0 & \cdots & 0 \\ 0 & A_2 I_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_J I_J \end{bmatrix}$$

where $\{I_j\}$ are identity matrices whose sizes are in concordance with $\{\vec{w}_j\}$. Now we can write (10) and (11) as one single linear system:

$$\begin{bmatrix} \Lambda + \Phi^T D_a \Phi & \Phi^T D_a \vec{1} \\ \vec{a}^T \Phi & \vec{a}^T \vec{1} \end{bmatrix} \begin{bmatrix} W \\ b \end{bmatrix} = \begin{bmatrix} \Phi^T D_a \vec{y} \\ \vec{a}^T \vec{y} \end{bmatrix} \tag{12}$$

So far we have not made any assumption about the form of the solution, thus this formulation could be directly applied to the original SVC. At this point, we introduce the parametric approximation, and we assume that the solution of the equation has a form

$$\vec{w}_j = \Psi_j^T \vec{\beta}_j \quad j = 1, \ldots, J \tag{13}$$

where $\Psi_j = [\vec{\phi}_j(\vec{c^j}_1) \cdots \vec{\phi}_j(\vec{c^j}_{Rj})]^T$. The bias term b of the original solution is kept. Now, we can write the solution of the system in terms of the parametric approximation:

$$W = \Psi^T \vec{\beta}$$

where

$$\Psi = \begin{bmatrix} \Psi_1 & 0 & \dots & 0 \\ 0 & \Psi_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Psi_J \end{bmatrix}$$

and $\vec{\beta} = [\vec{\beta}_1^T \cdots \vec{\beta}_J^T]^T$. This parametric solution is included in (12). Multiplying each side of (12) by

$$\begin{bmatrix} \Psi & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}$$

and grouping terms, we arrive to:

$$\begin{bmatrix} \Psi\Lambda\Psi^T + \Psi\Phi^T D_a \Phi\Psi^T & \Psi\Phi^T D_a \vec{1} \\ \vec{a}^T \Phi\Psi^T & \vec{a}^T \vec{1} \end{bmatrix} \begin{bmatrix} \vec{\beta} \\ b \end{bmatrix} = \begin{bmatrix} \Psi\Phi^T D_a \vec{y} \\ \vec{a}^T \vec{y} \end{bmatrix} \tag{14}$$

Products between matrices Φ and Ψ are inner products of vectors in the projected spaces, and can be written in their kernel forms:

$$\begin{bmatrix} I_\Psi + K^T D_a K & K^T D_a \vec{1} \\ \\ \vec{a}^T K & \vec{a}^T \vec{1} \end{bmatrix} \begin{bmatrix} \vec{\beta} \\ \\ b \end{bmatrix} = \begin{bmatrix} K^T D_a \vec{y} \\ \\ \vec{a}^T \vec{y} \end{bmatrix} \tag{15}$$

where

$$I_\Psi = \Psi \Lambda \Psi^T = \begin{bmatrix} A_1 \Psi_1 \Psi_1^T & 0 & \ldots & 0 \\ \\ 0 & A_2 \Psi_2 \Psi_2^T & \ldots & 0 \\ \\ \vdots & \vdots & \ddots & \vdots \\ \\ 0 & 0 & \ldots & A_J \Psi_J \Psi_J^T \end{bmatrix}$$

$$K = \Phi \Psi^T = \left[\Phi_1 \Psi_1^T \cdots \Phi_J \Psi_J^T \right]$$

Here, it is worth remarking that, due to the block-diagonal structure of matrices I_Ψ and Ψ, inner products between projected vectors corresponding to different mappings associated to different kernels k_j, k_l with $j \neq l$ never appear, i.e. all the inner products involve well defined projections and can be calculated via the corresponding kernel. In this sense, I_Ψ and K can be calculated with kernels using:

$$I_{\Psi j} = \delta_j \Psi_j \Psi_j^T; \; (I_{\Psi j})_{mn} = \delta_j k_j(\vec{c^j}_m, \vec{c^j}_n); \; m, n = 1, \ldots, R_j \tag{16}$$

$$K_j = \Phi_j \Psi_j^T; \quad (K_j)_{ik} = k_j(\vec{x}_i, \vec{c^j}_k); \quad i = 1, \ldots, N; k = 1, \ldots, R_j \tag{17}$$

Once we have calculated matrices I_Ψ and K, (15) can be solved via WLS-SVC to obtain the parametric approximation to the MK-SVC formulated in (12).

A growing scheme for the Gaussian MK-SVC case can be easily implemented based on the lines explained in Section 3.1. The idea consists on iteratively building the classifier incorporating centroids with different value of σ, according to the problem. In Figure 1, a schematic of the growing procedure is depicted. The machine is initialized with few centroids and with a high value of parameter σ, so that data are examined with a coarse resolution. The GSVC proceeds normally until the CE of the validation set shows that training has saturated. Then, σ is decreased in order to tackle in a deeper detail the parts of the classification boundary that are still badly learned. The value of σ is decreased after each saturation of the learning until no improvement is observed in the validation error. Moreover, from iteration to iteration there is no need to recalculate all the kernel matrices, but just to add the blocks corresponding to the new nodes. The algorithm, denominated MK-GSVC is summarized in Table 2.

3.3 Fast Classification with GSVC: FC-GSVC

As stated before, one of GSVC goals is to reduce the classifying cost. In what follows, we adopt the number of kernel evaluations per pattern (KEPP) as a measure for this cost, since it concentrates most of the computation effort. In this sense, the final machine built by GSVC is much faster than the QP-SVC one because it has a number of nodes (usually) much smaller than the number of support vectors, leading to a lower KEPP. Besides, GSVC provides a direct way to reduce the KEPP even further.

In [14] a method for sequential evaluation of SVMs that speeds up the classification is presented. The key point is that some regions of the input space (those lying far apart from the margin) can be correctly classified using only a reduced subset of the kernels. Similar schemes for fast classification can be implemented with GSVC (FC-GSVC) by sequentially computing $\{o_i\}, i = 0, \ldots, S-1$ (see Figure 1) as needed: if $o_j(\vec{x})$ gives enough evidence about the classification of pattern $\vec{x}$, the process stops and the pattern is classified according to $f(\vec{x}) = sign(o_j(\vec{x}))$; otherwise, $o_{j+1}(\vec{x})$ must be computed.

Now, we need a criterion to decide if o_j offers enough evidence to classify a pattern. We propose to apply a positive and a negative threshold associated to each $\{o_i\}$. Then, if $o_j(\vec{x})$ is above or below the corresponding thresholds we accept the classification. If no previous classification has been made, we classify the pattern with the GSVC output ($o_{S-1}(\vec{x})$). The thresholds are established in a way that guarantees that no training pattern correctly classified by GSVC is misclassified using the Fast Classification method. To accomplish this criterion, we fix the positive threshold for o_j to the maximum value of $o_j(\vec{x}_s)$, where $\{\vec{x}_s\}$ are all the training patterns belonging to class "-1" that are correctly classified by the GSVC. Negative thresholds are set in an analogous manner. All this involves little extra computational effort since $\{o_j(\vec{x})\}$ are already available as a result of the incremental traiing scheme.

It is worth remarking that, as the the growing procedure implies the addition (but not the modification) of centroids, there is no need to evaluate a number of kernels greater than the number of nodes of GSVC in any case, so FC-GSVC KEPP never exceeds that of GSVC.

We summarize FC-GSVC algorithm in Table 3 for a case in which we have used S steps to train the GSVC.

4 Experimental work

4.1 Benchmark datasets

This experiment is aimed at testing GSVC and MK-GSVC generalization capability and machine compactness. The datasets are selected from UCI machine learning repository [18]: Abalone (denominated *abalone* in the paper), Waveform Data Generator (*waveform*), Image Segmentation (*image*) and Diabetes (*diabetes*). We have also included a bidimensional synthetic problem proposed in [19] (*kwok*). Abalone has a training set of 2507 patterns and a test set of 1670, with inputs of 8 dimensions; waveform has inputs of 21 dimensions and the training and test sets sizes are of 4000 and 1000 patterns; image has 18 dimension inputs and the number of training and test patterns are 1848 and 462; diabetes has 8 dimensions, 576 training patterns and 192 test patterns; finally, kwok has a two dimension input space and its training and test data sets are formed by 500 and 10200 patterns.

We compare our schemes with SVMlight, an implementation of QP-SVC described in [9], and well known in the literature. We evaluate the classification error (CE) on the test set, as well as the test KEPP of the machine. Due to the non-deterministic nature of GSVC and MK-GSVC (random initialization of the machine and centroid selection), we provide average values over 20 trials, together with standard deviations. In these experiments, a 20% of the training data is reserved for the cross-validation that is needed to stop growth. Tables 4 and 5 show the results of applying these systems on the benchmark problems.

As for kernel parameter selection in GSVC (σ) and QP-SVC (g), we have used those corresponding to the best results when applying a five-fold cross-validation method. To do so, we test a few σ values in the range of 0.5 and 50. Later, we select the two best σ values and check again, by means of the five-fold procedure, ten values between them. The selected values are displayed in Table 6. For the growing algorithms, we also indicate the number of centroids per class (NCC) that are added at every step of the algorithm.

Notice that MK-GSVC is started in all problems with the same initial value of σ and NCC, avoiding their *a priori* estimation. Every time the growing algorithm decides to refine the detail of data analysis, the value of parameter σ is modified by multiplying its current value by a factor μ $(0 < \mu < 1)$. In these experiments μ was fixed to 0.7, since it was observed to work reasonably well, and the final machines obtained with this algorithm showed not to be very sensitive to this parameter.

We observe that GSVC outperforms SVMlight except for the *kwok* case. Besides, the number of kernel evaluations per pattern is, at least, one order of magnitude below (apart from *image* dataset). The application of MK-GSVC skips the estimation of kernel parameters, while still achieving results in terms of performance and machine size comparable to those of GSVC.

We have also applied our Fast Classification method to all the 20 GSVC machines trained for each dataset. The differences in performance are not significant, except for the *diabetes* dataset, maybe because in this case the number of training data is not large enough to make a good estimation of the thresholds. However, the KEPP reductions are very significant, ranging (apart from *diabetes*) from 66 % (*image*) to 73.3 % (*waveform*) with respect to GSVC. This reductions are made at no significant cost, since all the calculations to fix the thresholds and partial weights β_{ij}, as depicted in Figure 1, were carried out during the GSVC training.

4.2 Text Classification Experiment: the Reuters dataset

We report here text classification results of the GSVC algorithm (linear machine) on the Reuters database [20] using the ModApte Split as explained in [3], and using all word counts (apart from those in the stop-word list) as input, and we did not use stemming. As usual, we only used the 10 most populated categories. We detail the results in Table 7, where we indicate average Break Even Point (in percentage) between precision and recall measurements for every category as well as micro and macroaveraging [2] results.

It can be observed that the GSVC algorithm provides better results than SVMlight. The good performance of the GSVC algorithm is due to the compact representation capabilities underlying it, already discussed in [13][21]. In this sense, the cross-validation procedure incorporated in the GSVC framework seems to add extra generalization benefits to the SVMs. Furthermore, GSVC is able to achieve about 1 order of magnitude reduction of the machine size (GSVC obtains on average solutions with 150 nodes, whereas SVMlight obtains around 1000 nodes).

---

[2] Macroaverage is the arithmetic mean of the results in the ten categories, while microaverage is the mean with each result weighted by the number of relevant documents to the corresponding category.

5 Conclusions and Further Work

In this paper, we have presented an algorithm that iteratively grows a Support Vector Classifier (GSVC), in a problem-oriented form. GSVC algorithm is simple, efficient, and allows to control the trade-off between machine complexity and performance in terms of classification error. Experimental results in several benchmark problems point out that GSVC generalizes better than standard SVC, mostly due to its problem-oriented growing criterion, stopped by a crossvalidation procedure. These experiments also show significant reductions on the final machine size built by GSVC with respect to the original SVC.

Moreover, GSVC enables the design of a multi-kernel classifier, MK-GSVC, that has been used here to tackle classification tasks in a multi-resolution approach, its main advantage being that it skips the problem of *a priori* estimation of kernel hyperparameters. This MK-GSVC formulation also opens lines for further research in SVCs with mixtures of kernels.

In addition to this, the iterative nature of GSVC has been exploited to develop a method to evaluate the machine that speeds up its performance. Experimental results reveal that FC-GSVC achieves reductions of about a 65-75% in the number of kernel evaluations necessary to classify a pattern with respect to those of GSVC, at no extra cost during the training phase.

Ongoing research includes schemes for incremental learning, that would introduce adaptive elements in GSVC, such as node pruning or weight forgetting. Moreover, the scope of the applications is widening by the development of multicategory versions of the algorithm to deal with non-binary problems, as well as the development of highly efficient on-line versions to deal with huge datasets for Data Mining applications.

Acknowledgements

Authors are specially grateful to Dr. F. Pérez-Cruz and Mr. A. Caamaño-Fernández for their valuable comments and support during the realization of this work. We would also like to thank Dr. F. Valverde-Albacete and Dr. H. Molina-Bulla for their work in conducting the text classification experiments on the Reuters dataset.

References

[1] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (1998) 121 – 167.

[2] V. Vapnik, The nature of statistical learning theory, Springer-Verlag, New York, 1995.

[3] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proc. 10th European Conference on Machine Learning, Vol. 1, Chemnitz, Germany, 1998, pp. 137 – 142.

[4] J. Huang, X. Shao, H. Wechsler, Face pose discrimination using support vector machines (SVM), in: Proc. 14th International Conference on Pattern Recognition, (ICPR), Vol. 1, Brisbane, Queensland, Australia, 1998, pp. 154 – 156.

[5] M. Bonneville, J. Meunier, Y. Bengio, J. P. Souvy, Support vector machines

for improving the classification of brain PET images, in: Proc. of SPIE Medical Imaging Symposium, Vol. 3338, San Diego, CA, 1998, pp. 264 – 273.

[6] F. Pérez-Cruz, P. Alarcón-Diana, A. Navia-Vázquez, A. Artés-Rodríguez, Fast training of support vector classifiers, in: T. Leen, T. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems, Vol. 13, MIT Press, Cambridge MA, 2001, pp. 734 – 740.

[7] C. J. C. Burges, Simplified support vector decision rules, in: L. Saitta (Ed.), Proc. of the 13th Int. Conference on Machine Learning, Morgan Kaufmann, San Mateo CA, 1996, pp. 71 – 77.

[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] T. Joachims, Making large-scale SVM learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods – Support Vector Learning, MIT Press, Cambridge, MA, 1999, pp. 169 –184.

[10] A. J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: P. Langley (Ed.), Proc. of the 17th International Conference on Machine Learning, Morgan Kaufman, San Francisco CA, 2000, pp. 911 – 918.

[11] B. Schölkopf, Nonlinear component analysis in a kernel eigenvalue problem, Neural Computation 10 (1998) 1299–1319.

[12] E. Parrado-Hernández, J. Arenas-García, I. Mora-Jiménez, A. Navia-Vázquez, On problem-oriented kernel refining, submitted to Neurocomputing.

[13] A. Navia-Vázquez, F. Pérez-Cruz, A. Artés-Rodríguez, A. R. Figueiras-Vidal, Weighted least squares training of support vector classifiers leading to compact and adaptive schemes, IEEE Trans. Neural Networks 12 (5) (2001) 1047 – 1059.

[14] B. Schölkopf, A. J. Smola, Learning with kernels, MIT Press, Cambridge MA, 2002.

[15] B. Schölkopf, A. Smola, K. R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Tech. Rep. 44, Max-Planck-Institut für biologische Kybernetik (1996).

[16] 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, in: P. Levi, M. Schanz, R.-J. Ahlers, F. May (Eds.), Proc. 20 DAGM Symposium Mustererkennung, Vol. 1 of Informatik aktuell, Springer, Berlin, Germany, 1998, pp. 124 – 132.

[17] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119 – 139.

[18] C. L. Blake, C. J. Merz, UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, [http://www.ics.uci.edu/∼mlearn/MLRepository.html] (1998).

[19] J. T. Kwok, Moderating the output of support vector machine classifiers, IEEE Trans on Neural Networks 10 (5) (1999) 1018 –1031.

[20] D. D. Lewis, The Reuters-21578 Text Categorization Test Collection, available at [http://www.research.att.com/∼lewis/reuters21578.html].

[21] E. Parrado-Hernández, I. Mora-Jiménez, A. Navia-Vázquez, Growing support vector classifiers via architecture boosting, in: Proc. Learning'00, Leganés, Madrid, 2000.

BIOGRAPHICAL SKETCHES

E. Parrado-Hernández received the Telecommunication Engineer degree from Universidad de Valladolid, Spain, in 1999. At the present moment, he is a Ph.D. student in signal and data processing at the Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Spain.

His research interests include nonlinear processing, specially based on kernel methods, and its application to data mining and information retrieval.

I. Mora-Jiménez received the engineering degree in 1998 from the Universidad Politécnica de Valencia, Valencia, Spain. Since then she is pursuing the Ph.D. degree in artificial neural networks at the Universidad Carlos III de Madrid, Madrid, Spain.

Her main research interests include machine learning, data mining and artificial neural networks.

J. Arenas-García was born in Seville, Spain, in 1977. He received the Telecommunication Engineer degree in 2000 from Universidad Politécnica de Madrid (ranked number 1; National Award to graduation). He is currently pursuing the Ph.D. degree at the Department of Signal Theory and Communications, Universidad Carlos III de Madrid. His present research interests are focused in the fields of neural networks and learning theory.

A.R. Figueiras-Vidal received the Telecomm Engineer degree from Universidad Politécnica de Madrid, Spain, in 1973 and the Doctor degree in 1976 from Universidad Politécnica de Barcelona.

He is a Professor in Signal Theory and Communications at Universidad Carlos III de Madrid, and at the present moment, Head of its Department of Signal Theory and Communications. His research interests are digital signal processing, digital communications, neural networks, and learning theory.

In these subjects he has coauthored more than 200 international journal and conference papers.

Dr. Figueiras is a member of the Spain Academy of Engineering.

A. 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). He is now an Associate Professor at the Department of Communication Technologies, 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 multimedia processing, communications, data mining, knowledge management and teleeducation. He has (co)authored more than 20 international journal and conference papers in these areas.

FIGURE CAPTIONS

Fig. 1. GSVC architecture evolution after N steps of growing. Solid arrows: weight paths of the resulting classifier. Dashed arrows: weights of the partial machines, with signals O_i being the outputs before taking the sign. The machine is started with a wide σ (nodes in the top box) and it is grown until the stopping criterion is met. Then σ is decreased, the procedure is repeated for adjusting weights of the new partial machine (nodes in the middle and bottom boxes). The refining continues until no improvement of the performance is observed.

Table 1

Summary of the GSVC algorithm

0.-	Initialization
-	Pick randomly $\frac{M}{2}$ patterns from each class
-	Build initial kernel matrix K_0
-	Obtain the initial hyperplane and SVs $(\vec{\alpha}(0), \vec{\beta}(0))$ = WLS-SVC(K_0,$\vec{y}$)
	Loop n = 1,2,...
1.-	Find a new set of centroids to expand the architecture, $\{\vec{c}_k(n)\}$
2.-	Update the kernel matrix $(K_n)_{ik} = k(\vec{x}_i, \vec{c}_k(n))$ and build $K = [K_0, \ldots, K_n]$
3.-	Train the new architecture, $(\vec{\alpha}(n), \vec{\beta}(n))$ = WLS-SVC(K,$\vec{y}$)
4.-	Identify SVs and apply *shrinking*
5.-	Calculate the CE on the validation set
6.-	Calculate the number of changes from SV to non-SV and vice versa
7.-	Decide whether to stop or continue growing

Table 2

Summary of the MK-GSVC algorithm

0.-	Initialization	
-	Choose initial value of parameter $\sigma = \sigma_0$	
-	Pick randoly $\frac{M}{2}$ patterns from each class	
-	Build initial kernel matrix with σ_0 K_0	
-	Obtain the initial hyperplane and SVs $(\vec{\alpha}(0), \vec{\beta}(0))$ = WLS-SVC(K_0,$\vec{y}$)	
-	Iterate GSVC until saturation	
Loop n = 1,2,...		
1.-	Find a new value for parameter $\sigma = \sigma_n$	
	Loop m = 1,2,...	
	1.1.-	Find a new set of centroids to expand the architecture, $\{\vec{c}_k(n,m)\}$
	1.2.-	Update $(K_{nm})_{ik} = k_n(\vec{x}_i, \vec{c}_k(n,m))$ and build $K = [K_0, \dots, K_{nm}]$
	1.3.-	Update I_Ψ
	1.4.-	Train the new architecture, $(\vec{\alpha}(n,m), \vec{\beta}(n,m))$ = WLS-SVC(K,I_Ψ,$\vec{y}$)
	1.5.-	Identify SVs and apply *shrinking*
	1.6.-	Calculate the CE on the validation set
	1.7.-	Calculate the number of changes from SV to non-SV and vice versa
	1.8.-	Decide whether to continue growing with current σ or not
2.-	Decide whether to continue decreasing σ or stop	

Table 3

Fast Classification with GSVC. T_+ and T_- are two arrays with the $S-1$ positive and negative thresholds

1.-	Loop i = 1,2,...,S-2 or until classification
-	If $o_i(\vec{x}) > T_+(i)$, classify $\vec{x}$ as belonging to class +1.
-	If $o_i(\vec{x}) < T_-(i)$, classify $\vec{x}$ as belonging to class -1.
2.-	If $\vec{x}$ has not yet been classified, classify it with $o_{S-1}(\vec{x})$

Table 4

Test classification error obtained by GSVC, MK-GSVC, FC-GSVC and QP-SVC in several benchmark problems.

	Test CE (%)			
Problem	GSVC	MK-GSVC	FC-GSVC	QP-SVC
abalone	19.42±0.58	23.56±1.51	19.63±0.26	19.7
waveform	8.60±0.23	8.97±0.41	8.69±0.19	8.9
image	3.8±0.3	4 ± 0.7	3.8 ±0.6	3.9
diabetes	23.8± 1.7	22.8 ± 1.1	24.3 ± 1.2	25
kwok	12.07±0.23	12.89±1.03	12.14±0.22	11.8

Table 5

Test KEPP for GSVC, MK-GSVC, FC-GSVC and QP-SVC in several benchmark problems.

	Test KEPP			
Problem	GSVC	MK-GSVC	FC-GSVC	QP-SVC
abalone	52.40±15.38	85.56±32.33	17.76±5.05	1184
waveform	80.4±16.33	225.84±88.18	21.43±1.85	892
image	160±34.58	261.9±37.13	54.42±7.73	271
diabetes	18.74±4.07	71.6±18.72	8.52±2.57	295
kwok	33.16±10.29	126.0±38.88	8.97±1.72	122

Table 6

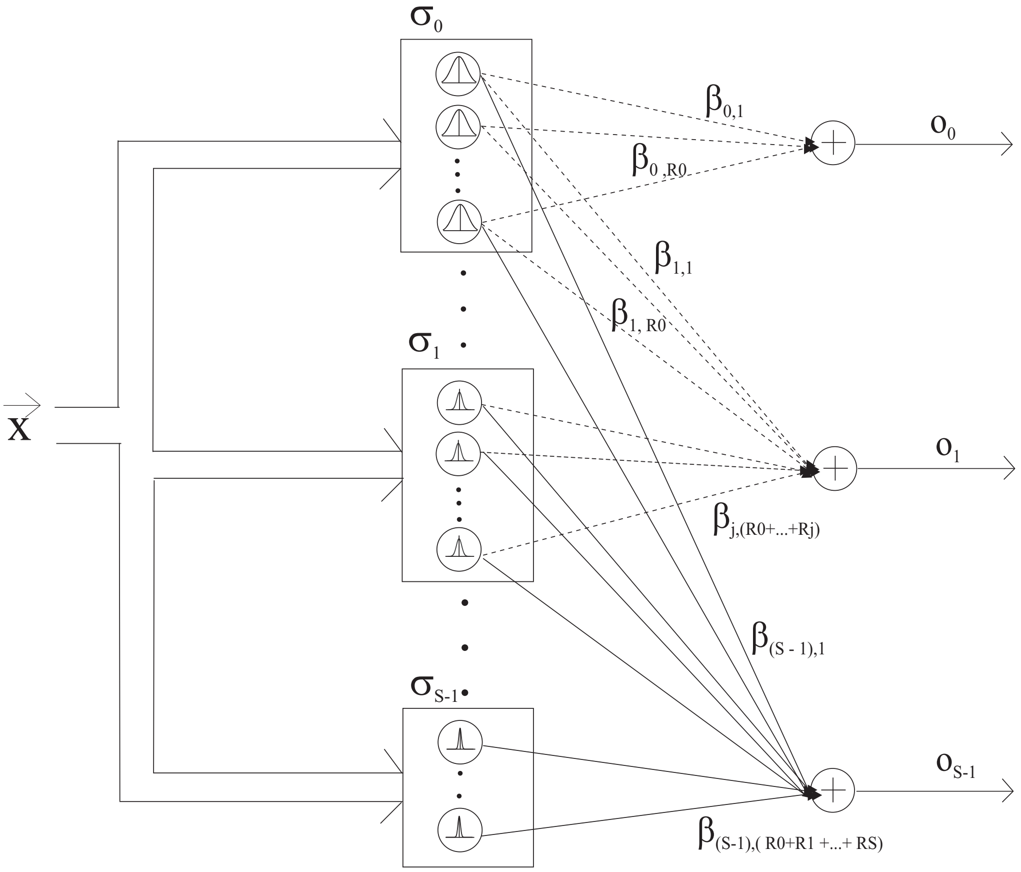
Parameters for GSVC, MK-GSVC and QP-SVC in the benchmark problems.

Problem	GSVC	MK-GSVC	QP-SVC
abalone	σ=1.1, NCC=4	σ_0=50, NCC=2	$g = 0.15$
waveform	σ=26, NCC=10	σ_0=50, NCC=2	$g = 0.0022$
image	σ=2.5, NCC=4	σ_0=50, NCC=2	$g = 0.1$
diabetes	σ=1.6, NCC=1	σ_0=50, NCC=2	$g = 0.2$
kwok	σ=3, NCC=4	σ_0=50, NCC=2	$g = 0.125$

Table 7

Comparison between SVMlight and GSVC trained without stemming in the text classification problem. The second column shows the number of relevant documents to each category within the test set. GSVC results are the average on ten different realizations.

Category	No. positive docs (test)	SVMlight	GSVC
earn	1088	97.5	**97.8**
acq	719	91.5	**94.8**
money-fx	180	67.1	**77.5**
grain	149	79.8	**87.6**
crude	189	83.6	**85.5**
trade	117	70.9	**81.2**
interest	133	59.5	**74.1**
ship	89	71.9	**72.5**
wheat	71	74.6	**80.1**
corn	56	73.2	**79.9**
MICROAVER.		87.3	**90.9**
MACROAVER.		76.9	**83.1**



σ0
σ1
σS-1
x
β0,1
β0 ,R0
β1,1
β1, R0
βj,(R0+...+Rj)
β(S - 1),1
β(S-1),(R0+R1 +...+ RS)
o0
o1
oS-1