Incremental and Decremental Learning for Linear Support Vector Machines

Enrique Romero, Ignacio Barrio, and Lluís Belanche

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. We present a method to find the exact maximal margin hyperplane for linear Support Vector Machines when a new (existing) component is added (removed) to (from) the inner product. The maximal margin hyperplane with the new inner product is obtained in terms of that for the old inner product, without re-computing it from scratch and the procedure is reversible. An algorithm to implement the proposed method is presented, which avoids matrix inversions from scratch. Among the possible applications, we find feature selection and the design of kernels out of similarity measures.

1 Introduction

Support Vector Machines (SVMs) are widely used tools for classification and regression problems based on margin maximization [1]. The standard formulation of SVMs uses a fixed data set to construct a linear combination of simple functions depending on the data. In many application domains, however, the number of examples or the number of variables may vary with time. In these cases, adaptive solutions that efficiently work in these changing environments are of great interest. Procedures for exact incremental (on-line) learning for Support Vector Machines (SVMs) have been introduced in [2,3] for classification and regression problems respectively. In [4,5], the dynamic adaptation of the kernel parameter was studied. In [6], the entire SVM solution path is obtained for every value of the regularization parameter.

In this work the situation is that of adapting the inner product for linear SVMs in an incremental manner. Specifically, suppose that we have obtained the maximal margin hyperplane of a linear SVM, and we would like to add/remove a new/existing component to/from the inner product. Can the maximal margin hyperplane with the new inner product be obtained in terms of that for the old inner product? A positive answer to this question would allow to add new variables to the data and obtain the linear SVM solution with these new variables without computing it from scratch. Analogously, existing variables could be removed from the SVM solution.

Similar to previous works [2,5], the main idea is the preservation of the Karush-Kuhn-Tucker (KKT) conditions to find the (exact) solution and guide the process. The solution will be updated in response to the perturbation induced by the addition/removal of the new/existing component to/from the inner product. The new/existing component will not be added/removed at once, but it will be added/removed in suitable increments that allow to control the migrations among the subsets of support vectors (vectors in the margin, violating the margin or exceeding the margin). For small increments (no migrations among the subsets of support vectors), the analysis leads to the solution of a linear equations system. When a migration occurs, the dimensions of the matrices involved in this linear equations system must be modified. The addition and removal of the new/existing component can be treated in the same way, so that the procedure is reversible. Direct applications include time-varying learning environments, feature selection and kernel design from arbitrary similarity measures.

2 Background

To fix notation, let $\mathcal{X} = \{(x_1, y_1), \ldots, (x_L, y_L)\}$ be a data set, and consider the classification task given by $\mathcal{X}$, where each instance $x_i = (x_i^1, \ldots, x_i^N) \in \mathbb{R}^N$ and $y_i \in \{-1, +1\}$. SVMs can be described as follows [1]: the input vectors are mapped into a (usually high-dimensional) inner product space through some non-linear mapping ϕ, chosen *a priori*. In this space (the *feature space*), an optimal separating hyperplane is constructed. By using a (positive definite) kernel function $K(u, v)$ the mapping can be implicit, since the inner product defining the hyperplane can be evaluated as $\langle\phi(u), \phi(v)\rangle = K(u, v)$ for every two vectors $u, v \in \mathbb{R}^N$. In the SVM framework, an optimal hyperplane means a hyperplane with maximal normalized margin for the examples of every class. The normalized margin is the minimum distance to the hyperplane.

When the data set is not linearly separable (neither in the input space nor in the feature space), some tolerance to noise is introduced in the model. Using Lagrangian theory, the maximal margin hyperplane for a binary classification problem given by a data set $\mathcal{X}$ is a linear combination of simple functions depending on the data: $f(x) = b + \sum_{i=1}^{L} y_i \alpha_i K(x_i, x)$, where $K(u, v)$ is a kernel function and the coefficients vector $(\alpha_i)_{i=1}^{L}$ is the (1-norm soft margin) solution of a constrained optimization problem in the dual space [1,2]:

$$\begin{array}{lll} \text{Minimize}(\boldsymbol{\alpha}, b) & W = \frac{1}{2} \sum_{i,j=1}^{L} y_i \alpha_i y_j \alpha_j K(x_i, x_j) + b \sum_{i=1}^{L} y_i \alpha_i - \sum_{i=1}^{L} \alpha_i & \\ \text{subject to} & 0 \leqslant \alpha_i \leqslant C \qquad i = 1, \ldots, L & \end{array} \tag{1}$$

for a certain constant C. Note this is not the standard formulation, since dual and primal variables are present at the same time. This allows to deal with the bias term in a similar way than with the rest of the variables. The regularization parameter C allows to control the trade-off between the margin and the training errors. The *hard margin* hyperplane is obtained with $C = \infty$.

An example is well classified if and only if its functional margin $y_i f(x_i)$ with respect to f is positive. The KKT conditions uniquely define the solution $\{\boldsymbol{\alpha}, b\}$ of (1):

$$g_i = \frac{\partial W}{\partial \alpha_i} = y_i f(x_i) - 1 \begin{cases} > 0 & \alpha_i = 0 \\ = 0 & 0 \leqslant \alpha_i \leqslant C \\ < 0 & \alpha_i = C \end{cases} \tag{2}$$

$$h = \frac{\partial W}{\partial b} = \sum_{i=1}^{L} y_i \alpha_i = 0 \tag{3}$$

Following [2,5], the training examples can be partitioned into three different categories: the set $\mathcal{S}$ of *margin support vectors* on the margin ($g_i = 0$), the set $\mathcal{E}$ of *error support vectors* violating the margin ($g_i < 0$) and the set $\mathcal{R}$ of *reserve vectors* exceeding the margin ($g_i > 0$).

An incremental algorithm for on-line training of SVMs for classification problems (1) was presented in [2]. When new examples are considered, the KKT conditions are preserved on all previously seen data and satisfied for the new data. This is done in a sequence of analytically computable steps. The whole procedure is reversible, thus allowing "decremental unlearning" and leave-one-out error estimation. In [5], the model is extended to adapt the SVM solution to changes in regularization and kernel parameters. A similar method for constructing ϵ-insensitive SVMs for regression problems is described in [3]. In [6], the entire path of the solutions of SVMs is obtained for every value of the regularization parameter in a similar way, exploting the fact that the coefficients α_i in the solution of (1) are piece-wise linear in C.

3 Incremental and Decremental Inner Product Learning for Linear SVMs

3.1 Problem Setting

Suppose that we are working with linear SVMs (wherein $K(x_i, x_j)$ is the *inner product* $\langle x_i, x_j \rangle = \sum_{k=1}^{m} x_i^k x_j^k$) and we would like to solve the following situations:

1. We have $m < N$ components in our inner product and we want to add a new component: $\langle x_i, x_j \rangle^{\text{new}} = \sum_{k=1}^{m+1} x_i^k x_j^k = \langle x_i, x_j \rangle + x_i^{m+1} x_j^{m+1}$.
2. We have $m+1 \leq N$ components in our inner product and we want to remove an existing component: $\langle x_i, x_j \rangle^{\text{new}} = \sum_{k=1}^{m} x_i^k x_j^k = \langle x_i, x_j \rangle - x_i^{m+1} x_j^{m+1}$.

The problem in both cases is to obtain the solution of (1) with the new inner product $\langle x_i, x_j \rangle^{\text{new}}$ in terms of that for the old one $\langle x_i, x_j \rangle$ (i.e., without recomputing it from scratch). A solution to these problems is described in the next sections.

3.2 Sketch of the Solution

We start from the solution $\{\boldsymbol{\alpha}, b\}$ of (1) with $\langle x_i, x_j \rangle$ (which satisfy the KKT conditions). In order to obtain the solution of (1) with the new inner product, we follow the strategy of computing the increments $\{\Delta\boldsymbol{\alpha}, \Delta b\}$ such that $\{\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}, b+\Delta b\}$ satisfy the KKT conditions of (1) with $\langle x_i, x_j \rangle^{\text{new}}$. The KKT conditions will be preserved by changing the parameters in response to the perturbation induced by the addition/removal of the new/existing component to/from the inner product. In this process, examples of the different categories ($\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$) may change their state. In order to control these changes, the new/existing component will not be added/removed at once, but added/removed in steps of a certain amplitude Δp such that it leads to the minimum number of category changes, which can be controlled. With the previous notation, the addition and removal of the new/existing component can be treated in the same way. During the process, the inner product will be

$$\langle x_i, x_j \rangle' = \langle x_i, x_j \rangle + p \cdot x_i^{m+1} x_j^{m+1}. \tag{4}$$

Initially, $p = 0$. When adding a new component, p must reach $p = 1$ with increments $\Delta p > 0$. When removing an existing component, p must reach $p = -1$ with increments $\Delta p < 0$. For every Δp (different at every step), the solution will be recalculated so that the KKT conditions (2,3) are preserved on all data (thus controlling the migrations among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$).

Therefore, for a given perturbation Δp, our objective is to determine the necessary changes in the solution of (1) and the migrations among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$ so that the KKT conditions are preserved on all data. Let us define

$$W' = \frac{1}{2} \sum_{i,j=1}^{L} y_i \alpha_i y_j \alpha_j \langle x_i, x_j \rangle' + b \sum_{i=1}^{L} y_i \alpha_i - \sum_{i=1}^{L} \alpha_i$$

with $\{\boldsymbol{\alpha}, b\}$ satisfying (2,3), and

$$W^+ = \frac{1}{2} \sum_{i,j=1}^{L} y_i \alpha_i^+ y_j \alpha_j^+ \langle x_i, x_j \rangle^+ + b^+ \sum_{i=1}^{L} y_i \alpha_i^+ - \sum_{i=1}^{L} \alpha_i^+$$

with $\alpha_i^+ = \alpha_i + \Delta\alpha_i$, $b^+ = b + \Delta b$ and $\langle x_i, x_j \rangle^+ = \langle x_i, x_j \rangle' + \Delta p \, x_i^{m+1} x_j^{m+1}$.

3.3 Calculation of the Increments

The increments $\{\Delta\boldsymbol{\alpha}, \Delta b\}$ can be computed as a function of Δp, as explained next. Two situations may arise, tackled in the rest of the section:

1. The solution changes, but no example changes its state among the different categories ($\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$). This is the typical situation when $|\Delta p|$ is small. In this case, only the coefficients of the support vectors and the bias term must be updated.

2. There exists one or more examples that migrate among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$. These changes can be controlled if Δp is chosen so that the minimum number of migrations occurs. In this case, $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$ must be updated, but recalculating the solution is not necessary.

1. Small $|\Delta p|$ (no migrations among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$). Suppose that $|\Delta p|$ is small enough so that no example changes its state: $\Delta\alpha_i = 0$ for every $i \notin \mathcal{S}$, $\alpha_i = 0$ for every $i \in \mathcal{R}$ and $\alpha_i = C$ for every $i \in \mathcal{E}$.

Since the new solution must satisfy the KKT conditions, we have:

$$g_i^+ = \frac{\partial W^+}{\partial \alpha_i^+} = y_i f^+(x_i) - 1 = y_i \left(\sum_{j=1}^{L} y_j \alpha_j^+ \langle x_i, x_j \rangle^+ + b^+ \right) - 1 = 0 \qquad \forall i \in \mathcal{S}$$

$$h^+ = \frac{\partial W^+}{\partial b^+} = \sum_{i=1}^{L} y_i \alpha_i^+ = 0.$$

Therefore, for every $i \in \mathcal{S}$ (recall that also $g_i' = \frac{\partial W'}{\partial \alpha_i} = 0$)

$$g_i^+ - g_i' = y_i \left(\sum_{j=1}^{L} y_j \alpha_j \Delta p \; x_i^{m+1} x_j^{m+1} \right) + y_i \left(\sum_{j=1}^{L} y_j \Delta\alpha_j \langle x_i, x_j \rangle^+ \right) + y_i \Delta b = 0.$$

Since no example changes its state, we have

$$\begin{aligned} 0 = & \, y_i \left(\sum_{j \in \mathcal{S}} y_j \alpha_j \Delta p \; x_i^{m+1} x_j^{m+1} \right) + y_i \left(\sum_{j \in \mathcal{E}} y_j C \Delta p \; x_i^{m+1} x_j^{m+1} \right) + \\ & y_i \left(\sum_{j \in \mathcal{S}} y_j \Delta\alpha_j \langle x_i, x_j \rangle^+ \right) + y_i \Delta b. \end{aligned}$$

In addition (recall that also $h' = \frac{\partial W'}{\partial b} = 0$),

$$h^+ - h' = \sum_{i=1}^{L} y_i \Delta\alpha_i = \sum_{i \in \mathcal{S}} y_i \Delta\alpha_i = 0.$$

The above analysis can be summarized as ($x_{\mathcal{S}_1}, \ldots, x_{\mathcal{S}_{L_\mathcal{S}}}$ are the examples in $\mathcal{S}$)

$$\left(\frac{\mathcal{Q}'}{\Delta p} + \mathcal{U} \right) \cdot \begin{pmatrix} \Delta b \\ \Delta\alpha_{\mathcal{S}_1} \\ \vdots \\ \Delta\alpha_{\mathcal{S}_{L_\mathcal{S}}} \end{pmatrix} = -\,\mathcal{U} \cdot \begin{pmatrix} 0 \\ \alpha_{\mathcal{S}_1} \\ \vdots \\ \alpha_{\mathcal{S}_{L_\mathcal{S}}} \end{pmatrix} - \mathcal{V} \tag{5}$$

where $\mathcal{Q}', \mathcal{U}$ are $(L_\mathcal{S}+1) \times (L_\mathcal{S}+1)$ symmetric matrices

$$\mathcal{Q}' = \begin{pmatrix} 0 & y_{\mathcal{S}_1} & \cdots & y_{\mathcal{S}_{L_\mathcal{S}}} \\ y_{\mathcal{S}_1} & \mathcal{Q}'_{\mathcal{S}_1\mathcal{S}_1} & \cdots & \mathcal{Q}'_{\mathcal{S}_1\mathcal{S}_{L_\mathcal{S}}} \\ \vdots & \vdots & \ddots & \vdots \\ y_{\mathcal{S}_{L_\mathcal{S}}} & \mathcal{Q}'_{\mathcal{S}_{L_\mathcal{S}}\mathcal{S}_1} & \cdots & \mathcal{Q}'_{\mathcal{S}_{L_\mathcal{S}}\mathcal{S}_{L_\mathcal{S}}} \end{pmatrix} \qquad \mathcal{U} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & \mathcal{U}_{\mathcal{S}_1\mathcal{S}_1} & \cdots & \mathcal{U}_{\mathcal{S}_1\mathcal{S}_{L_\mathcal{S}}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \mathcal{U}_{\mathcal{S}_{L_\mathcal{S}}\mathcal{S}_1} & \cdots & \mathcal{U}_{\mathcal{S}_{L_\mathcal{S}}\mathcal{S}_{L_\mathcal{S}}} \end{pmatrix} \tag{6}$$

$$\mathcal{Q}'_{\mathcal{S}_i\mathcal{S}_j} = y_{\mathcal{S}_i} y_{\mathcal{S}_j} \left\langle x_{\mathcal{S}_i}, x_{\mathcal{S}_j} \right\rangle' \qquad \mathcal{U}_{\mathcal{S}_i\mathcal{S}_j} = y_{\mathcal{S}_i} y_{\mathcal{S}_j} x_{\mathcal{S}_i}^{m+1} x_{\mathcal{S}_j}^{m+1} \tag{7}$$

and $\mathcal{V}$ is the vector

$$\mathcal{V} = \left(0, \mathcal{V}_{\mathcal{S}_1}, \cdots, \mathcal{V}_{\mathcal{S}_{L_\mathcal{S}}}\right)^t \qquad \mathcal{V}_{\mathcal{S}_i} = C\, y_{\mathcal{S}_i} x_{\mathcal{S}_i}^{m+1} \sum_{j \in \mathcal{E}} y_j x_j^{m+1}. \tag{8}$$

Therefore, given Δp, the increments of the parameters can be computed by solving the linear equations system (5). Note that, contrary to the situation in [2,5], the solution of (5) is not a linear function of Δp.

2. Large $|\Delta p|$ (controlling the migrations among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$). When $|\Delta p|$ is large, migrations among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$ may occur after solving (5). However, we can choose Δp such that the minimum number of migrations occurs. In this case several migrations may occur simultaneously, but an example can only migrate from its current set to a neighbor set (that is, between $\mathcal{S}$ and $\mathcal{E}$ or between $\mathcal{S}$ and $\mathcal{R}$). These migrations are determined by the KKT conditions:

1. From $\mathcal{E}$ to $\mathcal{S}$: One error support vector becomes a margin support vector. This happens when g_i (that was negative) becomes 0.
2. From $\mathcal{S}$ to $\mathcal{E}$: One margin support vector becomes an error support vector. This happens when its coefficient becomes C.
3. From $\mathcal{R}$ to $\mathcal{S}$: One reserve vector becomes a margin support vector. This happens when g_i (that was positive) becomes 0.
4. From $\mathcal{S}$ to $\mathcal{R}$: One margin support vector becomes a reserve vector. This happens when its coefficient becomes 0.

When a migration occurs, $\mathcal{Q}'$, $\mathcal{U}$ and $\mathcal{V}$ must be updated, since not only their components but also their dimension change. However, recomputing the solution is not necessary as a consequence of the migration: if a new support vector is inserted in $\mathcal{S}$, its coefficient remains the same (0 if migrated from $\mathcal{R}$ or C if migrated from $\mathcal{E}$); if an example is deleted from $\mathcal{S}$, its coefficient (which also equals 0 or C) will not vary until it is inserted in $\mathcal{S}$ again.

4 Implementation

4.1 Efficient Computation of the Linear Equations System

An efficient way to solve (5) would be desirable. This section describes an efficient way to update $(\mathcal{Q}'/\Delta p + \mathcal{U})^{-1}$ when:

1. $|\Delta p|$ is small enough so that no example changes its state.
2. Migrations among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$ occur.

1. Small $|\Delta p|$ (no changes in the dimension). Note that $\mathcal{U} = \boldsymbol{u} \cdot \boldsymbol{u}^t$, with $\boldsymbol{u} = \left(0, y_{\mathcal{S}_1} x_{\mathcal{S}_1}^{m+1}, \cdots, y_{\mathcal{S}_{L_\mathcal{S}}} x_{\mathcal{S}_{L_\mathcal{S}}}^{m+1}\right)^t$. Therefore, the inverse of $\mathcal{Q}'/\Delta p + \mathcal{U}$ can be efficiently computed by applying the Sherman-Morrison-Woodbury matrix inversion formula:

$$(\mathcal{Q}'/\Delta p + \mathcal{U})^{-1} = (\mathcal{Q}'/\Delta p + \boldsymbol{u} \cdot \boldsymbol{u}^t)^{-1} = \Delta p \left(\mathcal{Q}'^{-1} - \frac{\Delta p \ \mathcal{Q}'^{-1} \boldsymbol{u}\boldsymbol{u}^t \mathcal{Q}'^{-1}}{1 + \Delta p \ \boldsymbol{u}^t \mathcal{Q}'^{-1} \boldsymbol{u}} \right) \quad (9)$$

As a consequence, for every Δp such that $\mathcal{S}$ does not change, only $\mathcal{Q}'^{-1}$ is needed to compute $(\mathcal{Q}'/\Delta p + \mathcal{U})^{-1}$. Note that $\mathcal{Q}'^{-1}$ can also be updated in the same way when $p_{new} = p + \Delta p$, since $\mathcal{Q}'_{new} = \mathcal{Q}' + \Delta p \, \mathcal{U}$. Thus,

$$(\mathcal{Q}'_{new})^{-1} = \mathcal{Q}'^{-1} - \frac{\Delta p \ \mathcal{Q}'^{-1} \boldsymbol{u}\boldsymbol{u}^t \mathcal{Q}'^{-1}}{1 + \Delta p \ \boldsymbol{u}^t \mathcal{Q}'^{-1} \boldsymbol{u}} \quad (10)$$

2. Large $|\Delta p|$ (inserting/deleting examples in/from $\mathcal{S}$). When a new example is inserted/deleted in/from $\mathcal{S}$, matrix $\mathcal{Q}'^{-1}$ can be updated incrementally using block matrices techniques, thus avoiding the computation of $\mathcal{Q}'^{-1}$ from scratch, as explained next.

When a new margin support vector $x_{\mathcal{S}_{L_\mathcal{S}+1}}$ is inserted in $\mathcal{S}$, matrix $\mathcal{Q}'^{-1}$ is expanded as

$$\mathcal{Q}'^{-1} \leftarrow \begin{pmatrix} & & 0 \\ & \mathcal{Q}'^{-1} & \vdots \\ 0 & \cdots & 0 \end{pmatrix} + \frac{1}{\gamma} \begin{pmatrix} \mathcal{B} \\ 1 \end{pmatrix} \cdot \begin{pmatrix} \mathcal{B}^t & 1 \end{pmatrix} \quad (11)$$

where

$$\mathcal{B} = -\mathcal{Q}'^{-1} \cdot \mathcal{Q}'_{*L_\mathcal{S}+1}$$

$$\gamma = \mathcal{Q}'_{\mathcal{S}_{L_\mathcal{S}+1}\mathcal{S}_{L_\mathcal{S}+1}} - (\mathcal{Q}'_{*L_\mathcal{S}+1})^t \cdot \mathcal{Q}'^{-1} \cdot \mathcal{Q}'_{*L_\mathcal{S}+1}$$

with

$$\mathcal{Q}'_{*L_\mathcal{S}+1} = \left(y_{L_\mathcal{S}+1}, \mathcal{Q}'_{\mathcal{S}_1\mathcal{S}_{L_\mathcal{S}+1}}, \ldots, \mathcal{Q}'_{\mathcal{S}_{L_\mathcal{S}}\mathcal{S}_{L_\mathcal{S}+1}} \right)^t .$$

When an example $x_{\mathcal{S}_k}$ is deleted from $\mathcal{S}$, matrix $\mathcal{Q}'^{-1}$ is contracted as

$$\mathcal{Q}'^{-1}_{\mathcal{S}_i\mathcal{S}_j} \leftarrow \mathcal{Q}'^{-1}_{\mathcal{S}_i\mathcal{S}_j} - \frac{\mathcal{Q}'^{-1}_{\mathcal{S}_i\mathcal{S}_k} \mathcal{Q}'^{-1}_{\mathcal{S}_k\mathcal{S}_j}}{\mathcal{Q}'^{-1}_{\mathcal{S}_k\mathcal{S}_k}} \quad (12)$$

for every $\mathcal{S}_i, \mathcal{S}_j \in \{0, \mathcal{S}_1, \ldots, \mathcal{S}_{L_\mathcal{S}}\}$ such that $\mathcal{S}_i, \mathcal{S}_j \neq \mathcal{S}_k$. The index 0 refers to the first row/column.

```
AddRemoveComponent (data set $\mathcal{X}$, partitions $\{\mathcal{S},\mathcal{E},\mathcal{R}\}, \{\boldsymbol{\alpha}, b\}$ satisfying (2,3))
    $p \leftarrow 0$
    Compute $\mathcal{Q}'^{-1}$
    repeat
        Find the maximum $|\Delta p|$ (addition: $\Delta p \in (0, 1-p]$; removal: $\Delta p \in [-1-p, 0)$)
            such that, after solving (5) with (9), the number $M$ of migrations among
            $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$ is minimum
        Update $\mathcal{Q}'^{-1}$ with (10)
        $\{\boldsymbol{\alpha}, b\} \leftarrow \{\boldsymbol{\alpha}, b\} + \{\Delta\boldsymbol{\alpha}, \Delta b\}$, where $\{\Delta\boldsymbol{\alpha}, \Delta b\}$ is the solution of (5)
        if $M \neq 0$ then
            Update $\mathcal{Q}'^{-1}$ with (11) or (12) depending on whether it must be expanded
                or contracted (i.e., an example must be inserted/deleted in/from $\mathcal{S}$)
        end if
        $p \leftarrow p + \Delta p$
    until $|p| = 1$
end AddRemoveComponent
```

Fig. 1. An algorithm to add/remove a component to/from the inner product

4.2 A Complete Algorithm

The previous analysis leads to algorithm in figure 1. Several remarks are in order:

1. If succesive additions/removals have to be done, the first computation of $\mathcal{Q}'^{-1}$ is only necessary in the first step (for the rest, it can be a parameter).
2. The minimum number M of migrations among $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$ may be 0. In this case, $|p + \Delta p| = 1$ and the algorithm stops.
3. After updating $\mathcal{Q}'^{-1}$, recomputing the solution is not necessary (section 3.3).
4. After adding/removing one component, the values of $\{\mathcal{S},\mathcal{E},\mathcal{R}\}$, $\{\boldsymbol{\alpha}, b\}$ and $\mathcal{Q}'^{-1}$ can be used as parameters to a new call to **AddRemoveComponent**. Therefore, in the whole process of adding/removing iteratively several components to the inner product, only one matrix inversion has to be made from scratch (the one previous to the first call to the function).
5. To find the maximum $|\Delta p|$ we perform a binary (dichotomous) search in the corresponding real interval. The first cut-point is $\frac{1-p}{2}$ (addition) or $\frac{-1-p}{2}$ (removal). If no migrations occur, the search continues in the left subinterval, otherwise in the right subinterval. This process iterates until the value of $|\Delta p|$ converges to machine precision.

5 Applications

5.1 Feature Selection

A first and direct application of the algorithm described in section 4.2 is to perform feature selection with linear SVMs. Classical search algorithms for feature selection, like forward selection, backward elimination, or plus-l take-away-r

work by adding and removing features one at a time. Therefore, the algorithm described in section 4.2 can be easily used to add and remove input features in a prescribed way. Explicit feature selection search methods for linear SVMs (see [7], for example), can also benefit from this algorithm.

5.2 Basis Selection Guided by the Margin

Suppose that an explicit transformation of the input space is performed with a set of predefined basis functions $\Phi = \{\phi_j \mid \phi_j : \mathbb{R}^N \to \mathbb{R}\}_{j=1}^{M}$: for every $(x_i, y_i) \in \mathcal{X}$, consider the vector $(z_i, y_i) \in \mathbb{R}^M \times \mathbb{R}$, with $z_i = (\phi_1(x_i), \dots, \phi_M(x_i))$. The solution of a linear SVM in this new space would give a non-linear model in the original input space based on margin maximization. The problem, however, would be to select an appropiate subset of basis functions from Φ, since it seems clear that some of them may be useless for the problem at hand. In order to select this subset, a search process can be performed, which is equivalent to perform feature selection in the new space. The algorithm described in section 4.2 can be used to that end, adding and removing basis functions to/from the partially obtained solutions.

5.3 Kernel Out of a Similarity

A kernel function $K(x, y)$ can be seen as a form of *similarity measure* between objects x and y. There is a vast literature in the design of similarity measures in data analysis [8]. One of the main advantages is to be able to cope with data heterogeneity and special values (like missing values) with a clear semantics. However, the need to fulfill the positivity condition prevents their use with SVMs. Though there are some works on proving positivity for certain similarity measures [9], they are limited to certain simple measures and even then the property may be spoiled in presence of missing data.

A kernel can be defined from a similarity as follows. Given a similariry $s : X \times X \to \mathbb{R}$ and Z a finite subset of X, the function

$$K(x, y) = \sum_{z_i \in Z} s(x, z_i)\, s(y, z_i) \tag{13}$$

is a positive kernel. The kernel defined in (13) is the sum-product aggregation of the similarities between x, y by using their respective similarities to a third object $z_i \in Z \subseteq X$. The semantics is then that x, y are similar if both are consistently similar to a set of reference objects Z. We call the elements of this set Z the *kernel vectors*. This process can be expressed in terms of the data, if $X = \{x_1, \dots, x_L\}$ is the set of training vectors. In practice, the elements in Z (the kernel vectors) can be chosen to minimize (1) while keeping their number at a minimum, allowing more compact and computationally cheaper models. This can be done explicitly with a search algorithm that progressively adds/removes vectors to/from $Z \subseteq X$. In fact, this search process is similar to a feature selection search process.

6 Conclusions and Future Work

A method to find the exact maximal margin hyperplane for linear Support Vector Machines when a new (existing) component is added (removed) to (from) the inner product has been presented. The maximal margin hyperplane with the new inner product is obtained from the solution using the old inner product and the procedure is reversible. We have presented a full algorithm that implements the proposed method. Applications of the algorithm include time-varying learning environments wherein descriptive variables arrive one at a time, basis function selection with linear SVMs and kernel design from similarity measures, avoiding the need for positivity.

As future work, the method can be extended to non-linear SVMs where the objective would be to find solutions combining several kernels. In this new scenario, starting from the SVM solution with a certain kernel K_1, and given a function K_2 such that $K_1 + K_2$ is a kernel, we would like to obtain the solution with the new kernel $K^{\text{new}} = K_1 + K_2$. Similarly to the addition of components for linear kernels presented in this work, the new "component" K_2 can be added in suitable increments controlling the migrations among the support vectors.

Acknowledgments

This work was supported by the Consejo Interministerial de Ciencia y Tecnología (CICYT), under projects CGL2004-04702-C02-02 and TIN2006-08114.

References

1. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
2. Cauwenberghs, G., Poggio, T.: Incremental and Decremental Support Vector Machine Learning. In: Advances in Neural Information Processing Systems, vol. 12, pp. 409–415. MIT Press, Cambridge (2000)
3. Martín, M.: On-Line Support Vector Machine Regression. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 282–294. Springer, Heidelberg (2002)
4. Cristianini, N., Campbell, C., Shawe-Taylor, J.: Dynamically Adapting Kernels in Support Vector Machines. In: Advances in Neural Information Processing Systems, vol. 11, pp. 204–210. MIT Press, Cambridge (1999)
5. Diel, C., Cauwenberghs, G.: SVM Incremental Learning, Adaptation and Optimization. In: International Joint Conference on Neural Networks, vol. 4, pp. 2685–2690 (2003)
6. Hastie, T., Rosset, S., Tibshirani, R., Zhun, J.: The Entire Regularization Path for the Support Vector Machine. Journal of Machine Learning Research 5, 1391–1415 (2006)
7. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.N.: Gene Selection for Cancer Classification using Support Vector Machines. Machine Learning 46(1-3), 389–422 (2002)
8. Chandon, J.L., Pinson, S.: Analyse Typologique. Théorie et Applications. Masson (1981)
9. Gower, J.C., Legendre, P.: Metric and Euclidean Properties of Dissimilarity Coefficients. Journal of Classification 3, 5–48 (1986)