UNIVERSITÀ DEGLI STUDI FIRENZE

FLORE
Repository istituzionale dell'Università degli Studi di Firenze

A convergent hybrid decomposition algorithm model for SVM training

Questa è la Versione finale referata (Post print/Accepted manuscript) della seguente pubblicazione:

Original Citation:

A convergent hybrid decomposition algorithm model for SVM training / S. Lucidi; L. Palagi; A. Risi; M. Sciandrone. - In: IEEE TRANSACTIONS ON NEURAL NETWORKS. - ISSN 1045-9227. - STAMPA. - 20:(2009), pp. 1055-1060.

Availability:

This version is available at: 2158/388526 since:

Terms of use:

Open Access

La pubblicazione è resa disponibile sotto le norme e i termini della licenza di deposito, secondo quanto stabilito dalla Policy per l'accesso aperto dell'Università degli Studi di Firenze (https://www.sba.unifi.it/upload/policy-oa-2016-1.pdf)

Publisher copyright claim:

(Article begins on next page)

30 April 2024

J Optim Theory Appl (2009) 141: 107–126
DOI 10.1007/s10957-008-9489-9

Decomposition Algorithm Model for Singly Linearly-Constrained Problems Subject to Lower and Upper Bounds

C.J. Lin · S. Lucidi · L. Palagi · A. Risi · M. Sciandrone

Published online: 13 January 2009
© Springer Science+Business Media, LLC 2009

Abstract Many real applications can be formulated as nonlinear minimization problems with a single linear equality constraint and box constraints. We are interested in solving problems where the number of variables is so huge that basic operations, such as the evaluation of the objective function or the updating of its gradient, are very time consuming. Thus, for the considered class of problems (including dense quadratic programs), traditional optimization methods cannot be applied directly. In this paper, we define a decomposition algorithm model which employs, at each iteration, a descent search direction selected among a suitable set of sparse feasible directions. The algorithm is characterized by an acceptance rule of the updated point which on the one hand permits to choose the variables to be modified with a certain degree of freedom and on the other hand does not require the exact solution of any subproblem. The global convergence of the algorithm model is proved by assuming that the objective function is continuously differentiable and that the points of the level set have at least one component strictly between the lower and upper bounds. Numerical results on large-scale quadratic problems arising in the training of support vector machines show the effectiveness of an implemented decomposition scheme derived from the general algorithm model.

Communicated by P. Tseng.

C.J. Lin
Department of Computer Science, National Taiwan University, Taipei, Taiwan

S. Lucidi · L. Palagi
Dipartimento di Informatica e Sistemistica "Antonio Ruberti", University of Rome "La Sapienza", Rome, Italy

A. Risi
Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", CNR, Rome, Italy

M. Sciandrone (✉)
Dipartimento di Sistemi e Informatica, University of Florence, Florence, Italy
e-mail: sciandro@dsi.unifi.it

Keywords Large scale optimization · Decomposition methods · Support vector machines

1 Introduction

Let us consider the problem

$$\min \quad f(x), \tag{1a}$$

$$\text{s.t.} \quad a^T x = b, \tag{1b}$$

$$l \le x \le u, \tag{1c}$$

where $x \in R^n$, $f : R^n \to R$ is a continuously differentiable function and $a, l, u \in R^n$, with $l < u$, $b \in R$. We allow the possibility that some of the variables are unbounded by permitting both $l_i = -\infty$ and $u_i = \infty$ for some $i \in \{1, \dots, n\}$. Moreover, we assume, without loss of generality that $a_i \neq 0$ for all $i = 1, \dots, n$, thought our approach can be extended with minor modifications to include the case where $a_i = 0$ for some i. We focus the attention on large dimensional problems. We suppose that the dimension n is so large that traditional optimization methods cannot be directly employed since basic operations, such as the updating of the gradient or the evaluation of the objective function, are too time consuming. There are many real applications that can be modelled by optimization problems of the form (1). For instance, optimal control problems, portfolio selection problems, traffic equilibrium problems, multicommodity network flow problems (see, e.g., Refs. [1–4]) are specific instances of (1). Further, a continuous formulation of a classical problem in graph theory, the maximum clique problem, is formulated as (1) with indefinite quadratic function (see, e.g., Ref. [5]). Moreover, an important machine learning methodology, called support vector machine (SVM) (Ref. [6]), leads to huge problems of the form (1) with quadratic objective function. Different kinds of algorithms have been developed for large dimensional SVM training problems. Among them there are interior point methods (Ref. [7]), semismooth algorithms (Ref. [8]), methods based on unconstrained smooth reformulations (Refs. [9, 10]), active set methods (Refs. [11, 12]), projected gradient methods (Ref. [13]), decomposition methods (see, e.g., Refs. [14–17]). Here we focus our attention on a decomposition-based approach which, involving at each iteration the updating of a small number of variables, is suitable for large problems with dense Hessian matrix. In a general decomposition framework, at the generic iteration k the components of the current feasible point x^k are partitioned into two subsets W^k, usually called *working set*, and $\overline{W}^k = \{1, \dots, l\} \setminus W^k$ (for notational convenience from now on we omit the dependence of $W, \overline{W}$ on k). The variables corresponding to $\overline{W}$ are unchanged at the new iteration, while the components corresponding to W are set equal to those of a stationary point $x_W^\star$ of the subproblem

$$\min \quad f(x_W, x_{\overline{W}}^k), \tag{2a}$$

$$\text{s.t.} \quad a_W^T x_W = -a_{\overline{W}}^T x_{\overline{W}}^k + b, \tag{2b}$$

$$l_W \le x_W \le u_W. \tag{2c}$$

Thus, the new iterate is

$$x^{k+1} = \left(x_W^{k+1}, x_{\overline{W}}^{k+1}\right) = \left(x_W^*, x_{\overline{W}}^k\right).$$

In order to guarantee the global convergence of a decomposition method the working set selection cannot be arbitrary but must satisfy suitable rules (see, e.g., Refs. [14, 18, 19]). Up to our knowledge, the proposed convergent decomposition algorithms (except the method presented in Ref. [18]) and their convergence analysis are based on the assumption that, at each iteration, the computed point x_W^* exactly satisfies the optimality conditions of the generated subproblem (2). This can be a strong requirement, for instance whenever a stationary point of (2) cannot be analytically determined, but an iterative method must be necessarily applied. Then, the aim of the paper is the definition of a general decomposition algorithm model where, from one hand, the above requirement is removed, from the other, a degree of freedom in the choice of the variables to be updated at any iteration is introduced. In particular, the general scheme that will be defined is characterized by the following features.

– At each iteration, a finite subset of sparse feasible directions having only two nonzero elements is defined, provided that the current point has at least one component where the box constraints are not active. This set has cardinality $O(n)$, contains the generators of the set of feasible directions and has a structure well-suited for large scale problems.
– In the set mentioned above, a suitable direction is selected and an inexact line search along it is performed by means of an Armijo-type method; in this way a new candidate point is obtained without the need of computing an exact solution of a subproblem.
– The candidate point provides a reference value for the objective function and a new arbitrary iterate can be accepted if the corresponding function value does not exceed the reference value previously determined.
– The global convergence of the algorithm is proved by assuming that the objective function is continuously differentiable and that the points of the level set have at least one component where the box constraints are not active.

The paper is organized as follows. In Sect. 2, we introduce some basic notation and preliminary technical results. In Sect. 3 we state theoretical properties of a special class of feasible directions having only two nonzero components and characterizing our decomposition approach. In Sect. 4 we recall the well-known Armijo-type algorithm and its properties. In Sect. 5 we describe the decomposition algorithm model, whose theoretical properties are analyzed in Sect. 6. The numerical results obtained on large-scale SVM training problems in comparison with LIBSVM of Ref. [21] are reported in Sect. 7. Finally, in Appendix A we prove a technical result concerning polyhedral sets which is used in our convergence analysis, and in Appendix B we identify a class of SVM training problems which satisfy the assumption used in Sect. 6 to guarantee the global convergence of the proposed algorithm.

2 Notation and Preliminary Results

In this section, we introduce some basic notation and definitions, and we report some results proved in Ref. [19] that will be used in the sequel. Given a vector $x \in R^n$ and an index set $W \subseteq \{1, \dots, n\}$, we have already introduced the notation $x_W \in R^{|W|}$ to indicate the subvector of x made up of the component x_i with $i \in W$. Given a set $Y = \{y^1, \dots, y^m\} \subset R^n$, we indicate by cone$(Y)$ the convex cone of Y defined as follows:

$$\text{cone}(Y) = \left\{ y \in R^n : \ y = \sum_{h=1}^{m} \mu_h y^h, \ \mu_h \geq 0, h = 1, \dots, m \right\}.$$

We indicate by $\mathcal{F}$ the *feasible set* of Problem (1), namely

$$\mathcal{F} = \{x \in R^n : a^T x = b, l \leq x \leq u\}.$$

For every feasible point x, we denote the sets of indices of active (lower and upper) bounds as follows:

$$L(x) = \{i : x_i = l_i\}, \qquad U(x) = \{i : \ x_i = u_i\}.$$

The set of the *feasible directions* at a point $x \in \mathcal{F}$ is the cone

$$D(x) = \{d \in R^n : a^T d = 0, d_i \geq 0, \forall i \in L(x), \text{ and } d_i \leq 0, \forall i \in U(x)\}.$$

We say that a feasible point x^* is a *stationary point* of Problem (1) if

$$\nabla f(x^*)^T d \geq 0, \quad \text{for all } d \in D(x^*).$$

Since the constraints of Problem (1) are linear, we have that a feasible point $x^\star$ is a stationary point of Problem (1) if and only if the Karush-Kuhn-Tucker (KKT) conditions are satisfied, i.e. a scalar λ^* exists such that

$$(\nabla f(x^*))_i + \lambda^* a_i \begin{cases} \geq 0, & \text{if } i \in L(x^*), \\ \leq 0, & \text{if } i \in U(x^*), \\ = 0, & \text{if } i \notin L(x^*) \cup U(x^*). \end{cases}$$

The KKT conditions can be written in a different form. To this aim, the sets L and U can be split in L^-, L^+, and U^-, U^+ respectively, where

$$L^-(x) = \{i \in L(x) : a_i < 0\}, \qquad L^+(x) = \{i \in L(x) : a_i > 0\},$$
$$U^-(x) = \{i \in U(x) : a_i < 0\}, \qquad U^+(x) = \{i \in U(x) : a_i > 0\}.$$

We report the KKT conditions in the following proposition.

Proposition 2.1 (Optimality Conditions) *A point $x^* \in \mathcal{F}$ is a stationary point of Problem* (1) *if and only if there exists a scalar λ^* satisfying*

$$\lambda^* \geq -\nabla f(x^*)_i / a_i, \quad \forall i \in L^+(x^*) \cup U^-(x^*), \tag{3a}$$

$$\lambda^* \leq -\nabla f(x^*)_i / a_i, \quad \forall i \in L^-(x^*) \cup U^+(x^*), \tag{3b}$$

$$\lambda^* = -\nabla f(x^*)_i / a_i, \quad \forall i \notin L(x^*) \cup U(x^*). \tag{3c}$$

In correspondence to a feasible point x, we introduce the index sets

$$R(x) = L^+(x) \cup U^-(x) \cup \{i : l_i < x_i < u_i\}, \tag{4a}$$

$$S(x) = L^-(x) \cup U^+(x) \cup \{i : l_i < x_i < u_i\}. \tag{4b}$$

Proposition 2.2 *A feasible point x^* is a stationary point of Problem* (1) *if and only if, for any pair of indices (i, j), with $i \in R(x^*)$ and $j \in S(x^*)$, we have*

$$\nabla f(x^*)_i / a_i \geq \nabla f(x^*)_j / a_j. \tag{5}$$

Proposition 2.3 *Let $\{x^k\}$ be a sequence of feasible points convergent to a point $\bar{x}$. Then, for sufficiently large values of k, we have*

$$R(\bar{x}) \subseteq R(x^k) \quad \textit{and} \quad S(\bar{x}) \subseteq S(x^k).$$

3 Sets of Sparse Feasible Directions

In this section, we consider a special class of feasible directions having only two nonzero components and we show their important properties that will be employed in the decomposition approach presented later. Given $i, j \in \{1, \dots, n\}$, with $i \neq j$, we indicate by $d^{i,j}$ a vector belonging to R^n such that

$$d_h^{i,j} = \begin{cases} 1/a_i, & \text{if } h = i, \\ -1/a_j, & \text{if } h = j, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Given $x \in \mathcal{F}$ and the corresponding index sets $R(x)$ and $S(x)$, we indicate by $D_{RS}(x)$ the set of directions $d^{i,j}$ with $i \in R(x)$ and $j \in S(x)$, namely

$$D_{RS}(x) = \bigcup_{\substack{i \in R(x) \\ j \in S(x) \\ i \neq j}} d^{i,j}. \tag{7}$$

The following proposition was proved in Ref. [15].

Proposition 3.1 *Let $\hat{x}$ be a feasible point. For each pair $i \in R(\hat{x})$ and $j \in S(\hat{x})$, the direction $d^{i,j} \in R^n$ is a feasible direction at $\hat{x}$, i.e. $d \in D(\hat{x})$.*

The result stated below follows immediately from Proposition 2.2.

Proposition 3.2 *A feasible point x^* is a stationary point of Problem* (1) *if and only if*

$$\nabla f(x^*)^T d^{i,j} \geq 0 \quad \forall d^{i,j} \in D_{RS}(x^*). \tag{8}$$

The next proposition shows that for any feasible point x the set $D_{RS}(x)$ contains feasible directions and the generators of $D(x)$.

Proposition 3.3 *Given $\bar{x} \in \mathcal{F}$, we have*

$$D_{RS}(\bar{x}) \subseteq D(\bar{x}), \tag{9}$$

$$\text{cone}\{D_{RS}(\bar{x})\} = D(\bar{x}). \tag{10}$$

Proof Condition (9) is a consequence of Proposition 3.1. In order to prove (10), using (9), we must show that $d \in D(\bar{x})$ implies $d \in \text{cone}\{D_{RS}(\bar{x})\}$. Assume by contradiction that the thesis is false, so that, there exists a vector $\bar{d} \in D(\bar{x})$ such that $\bar{d} \notin \text{cone}\{D_{RS}(\bar{x})\}$. Hence, we have that the linear system

$$\bar{d} = \sum_{h=1}^{|D_{RS}(\bar{x})|} \mu_h d^h,$$
$$\mu_h \geq 0, \quad h = 1, \dots, |D_{RS}(\bar{x})|,$$

has no solution, where $d^h \in D_{RS}(\bar{x})$. Using Farkas's lemma, we have that there exists a vector $c \in R^n$ such that

$$c^T d^h \geq 0, \quad \forall d^h \in D_{RS}(\bar{x}), \tag{11}$$

$$c^T \bar{d} < 0. \tag{12}$$

Now, consider the linear function $F(x) = c^T x$ and the convex problem

$$\begin{aligned} \min \quad & F(x) = c^T x, && \text{(13a)} \\ \text{s.t.} \quad & a^T x = b, && \text{(13b)} \\ & l \leq x \leq u. && \text{(13c)} \end{aligned}$$

Condition (11) can be written as

$$\nabla F(\bar{x})^T d^h \geq 0, \quad \forall d^h \in D_{RS}(\bar{x}). \tag{14}$$

Using (14) and Proposition 3.2, we get that $\bar{x}$ is an optimal solution of Problem (13). On the other hand, $\bar{d}$ is a feasible direction at $\bar{x}$, and by (12) we have $\nabla F(\bar{x})^T \bar{d} < 0$. Then, $\bar{d}$ is a feasible descent direction at $\bar{x}$, and this contradicts the fact that $\bar{x}$ is an optimal solution of Problem (13). □

We observe that the set $D_{RS}(\bar{x})$ has cardinality, depending on $R(\bar{x})$ and $S(\bar{x})$, which is in the worst case $O(n^2)$. Now, under a suitable condition on the feasible point $\bar{x}$, we show that it can be easily defined a set of feasible directions containing, as $D_{RS}(\bar{x})$, the generators of $D(\bar{x})$, but having cardinality $O(n)$. In particular, let $\bar{x}$ be a feasible point with at least one "free" component, i.e. such that

$$l_h < \bar{x}_h < u_h, \tag{15}$$

for some index $h \in \{1, \dots, n\}$. In correspondence to a point $\bar{x}$ satisfying (15), we define the following set of directions:

$$D^h(\bar{x}) = \{d^{i,h} \in R^n : i \in R(\bar{x})\} \cup \{d^{h,j} \in R^n : j \in S(\bar{x})\}. \tag{16}$$

Remark 3.1 We note that $n - 1 \le |D^h(\bar{x})| \le 2(n-1)$. In fact in correspondence to each "free" index $t \ne h$, the set $D^h(\bar{x})$ must include the direction $d^{t,h}$ as $t \in R(\bar{x})$, and the direction $d^{h,t}$ as $t \in S(\bar{x})$. Hence if $R(\bar{x}) \cap S(\bar{x}) = \{1, \dots, n\}$, then $|D^h(\bar{x})| = 2(n-1)$. Conversely, for each index $t \notin R(\bar{x}) \cap S(\bar{x})$ there exists only one element of $D^h(\bar{x})$. Therefore, if $R(\bar{x}) \cap S(\bar{x}) = \{h\}$, then $|D^h(\bar{x})| = n - 1$.

As an example concerning the definition of $D^h(\bar{x})$, consider Problem (1) with $n = 4$, $l = (0, 0, 0, 0)^T$, $u = (3, 3, 3, 3)^T$, and $a = (1, -1, 1, -1)^T$. Let $\bar{x} = (0, 1, 3, 2)^T$ be a feasible point, so that $R(\bar{x}) = \{1, 2, 4\}$, $R(\bar{x}) \cap S(\bar{x}) = \{2, 4\}$, $S(\bar{x}) = \{2, 3, 4\}$. Setting $h = 2$, we obtain

$$D^2(\bar{x}) = \left\{ \begin{pmatrix} 1/a_1 \\ -1/a_2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1/a_2 \\ -1/a_3 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1/a_2 \\ 0 \\ -1/a_4 \end{pmatrix}, \begin{pmatrix} 0 \\ -1/a_2 \\ 0 \\ 1/a_4 \end{pmatrix} \right\}.$$

In the next proposition, we prove that the set $D^h(\bar{x})$ defined in (16) is indeed a subset of feasible directions and further that it contains the generators of the set of feasible directions $D(\bar{x})$.

Proposition 3.4 *Let $\bar{x} \in \mathcal{F}$ and let h be an index such that $l_h < \bar{x}_h < u_h$. Then, we have*

$$D^h(\bar{x}) \subseteq D(\bar{x}), \tag{17}$$

$$\text{cone}\{D^h(\bar{x})\} = D(\bar{x}), \tag{18}$$

where $D^h(\bar{x})$ is defined by (16).

Proof Note that, by assumption, the index h is such that $h \in R(\bar{x}) \cap S(\bar{x})$. Consider any $i \in R(\bar{x})$ (or any $j \in S(\bar{x})$) such that $d^{i,h} \in D^h(\bar{x})$ (or $d^{h,j} \in D^h(\bar{x})$). Then, $d^{i,h}$ ($d^{h,j}$) is such that $i \in R(\bar{x})$ and $h \in S(\bar{x})$ ($h \in R(\bar{x})$ and $j \in S(\bar{x})$), so that, condition (17) follows from Proposition 3.1.

In order to prove (18), using Proposition 3.3, it is sufficient to show that

$$\text{cone}\{D^h(\bar{x})\} = \text{cone}\{D_{RS}(\bar{x})\}. \tag{19}$$

Given any $d^{i,j} \in D_{RS}(\bar{x})$, we can write

$$d^{ij} = d^{ih} + d^{hj}. \tag{20}$$

On the other hand, by definition of $D^h(\bar{x})$, we have necessarily $d^{ih}, d^{hj} \in D^h(\bar{x})$, and hence (18) follows from (20) and (19). □

As a consequence of Propositions 3.2 and 3.4, we get directly the following result.

Proposition 3.5 *Let $\bar{x} \in \mathcal{F}$ be a point such that* (15) *holds and let h be an index such that $l_h < \bar{x}_h < u_h$. The feasible point $\bar{x}$ is a stationary point of Problem* (1) *if and only if*

$$\nabla f(\bar{x})^T d \geq 0, \quad \forall d \in D^h(\bar{x}), \tag{21}$$

where $D^h(\bar{x})$ is defined by (16).

4 Armijo-Type Line Search Algorithm

In this section, we describe the well-known Armijo-type line search along a feasible direction (see, e.g., Ref. [20]). The procedure will be used in the decomposition method presented in the next section. We state also some theoretical results useful in our convergence analysis. Let d^k be a feasible direction at $x^k \in \mathcal{F}$. We denote by $\beta_{\mathcal{F}}^k$ the maximum feasible steplength along d^k, namely $\beta_{\mathcal{F}}^k$ satisfies

$$l \leq x^k + \beta d^k \leq u, \quad \text{for all } \beta \in [0, \beta_{\mathcal{F}}^k],$$

and (since $-\infty \leq l < u \leq \infty$) we have that either $\beta_{\mathcal{F}}^k = +\infty$ or at least an index $i \in \{1, \dots, n\}$ exists such that

$$x_i^k + \beta_{\mathcal{F}}^k d_i^k = l_i \quad \text{or} \quad x_i^k + \beta_{\mathcal{F}}^k d_i^k = u_i.$$

Let β_u be a positive scalar and set

$$\beta^k = \min\{\beta_{\mathcal{F}}^k, \beta_u\}. \tag{22}$$

Assumption 4.1 Assume that $\{d^k\}$ is a sequence of feasible search directions such that

(a) for all k, we have $\|d^k\| \leq M$ for a given number $M > 0$;
(b) for all k, we have $\nabla f(x^k)^T d^k < 0$.

An Armijo-type line search algorithm is described below.

Algorithm 4.1 (Armijo-Type Line Search ALS(x^k, d^k, β^k))

```
Data: Given $\alpha > 0$, $\delta \in (0,1)$, $\gamma \in (0,1/2)$ and the initial stepsize $\alpha^k = \min\{\beta^k, \alpha\}$.
Step 1. Set $\lambda = \alpha^k$, $j = 0$.
Step 2. If
            $f(x^k + \lambda d^k) \leq f(x^k) + \gamma\lambda\nabla f(x^k)^T d^k$        (23)
        then set $\lambda^k = \lambda$ and stop.
Step 3. Set $\lambda = \delta\lambda$, $j = j + 1$ and go to Step 2.
```

The properties of Algorithm ALS are reported in the next proposition, whose proof can be obtained by repeating the reasonings used in Ref. [20] to prove Proposition 1.2.1.

Proposition 4.1 *Let $\{x^k\}$ be a sequence of points belonging to the feasible set $\mathcal{F}$ and let $\{d^k\}$ be a sequence of search directions satisfying Assumption 4.1. Then*:

(i) *For each k, Algorithm* ALS *determines, in a finite number of iterations, a scalar λ^k such that condition (23) holds, i.e.,*

$$f(x^k + \lambda^k d^k) \le f(x^k) + \gamma \lambda^k \nabla f(x^k)^T d^k. \tag{24}$$

(ii) *If $\{x^k\}$ converges to $\bar{x}$ and*

$$\lim_{k\to\infty} (f(x^k) - f(x^k + \lambda^k d^k)) = 0, \tag{25}$$

then we have

$$\lim_{k\to\infty} \beta^k \nabla f(x^k)^T d^k = 0, \tag{26}$$

where β^k is given by (22).

5 Decomposition Algorithm Model

In this section we describe a decomposition algorithm based on the employment of the feasible directions having only two nonzero components. The results stated in Sect. 3 show that, given any feasible point x^k, the set $D(x^k)$ of feasible directions at x^k can be generated by finite sets of directions with only two nonzero components. In particular, Proposition 3.3 shows that the set $D_{RS}(x^k)$ contains the generators of $D(x^k)$, while Proposition 3.4 gives the same result for the set $D^{i(k)}(x^k)$ under the assumption that $l_i < x_i^k < u_i$ for some index $i = i(k)$. Thus both the sets $D_{RS}(x^k)$ and $D^i(x^k)$ contain "sufficient" information. In the worst case $D_{RS}(x^k)$ has $n(n-1)/2$ elements, while $D^i(x^k)$ is made of $2(n-1)$ vectors. Therefore, in order to develop a feasible direction-based approach for large dimensional problems, it appears to be more appropriate to exploit a set of directions of the form $D^i(x^k)$. More in particular, we illustrate the strategy underlying our approach. In the case that the current feasible point x^k is such that (15) is verified, we identify a "suitable" index $i(k)$ (we highlight the dependence on the iteration counter k) corresponding to a "free" component $x_{i(k)}^k$, and we consider the following problem:

$$\min_{\beta\in R, d\in R^n} \quad \beta \nabla f(x^k)^T d, \tag{27a}$$

$$\text{s.t.} \quad 0 \le \beta \le \beta_u, \tag{27b}$$

$$x^k + \beta d \in \mathcal{F}, \tag{27c}$$

$$d \in D^{i(k)}(x^k), \tag{27d}$$

where β_u is a given positive scalar and

$$D^{i(k)}(x^k) = \{d^{h,i(k)} \in R^n : h \in R(x^k)\} \cup \{d^{i(k),h} \in R^n : h \in S(x^k)\}. \tag{28}$$

It is easy to see that problem (27) admits a solution. The rationale underlying the definition of problem (27) is that of identifying, among the generators of $D(x^k)$ having only two nonzero components, a feasible direction which locally may produce a sufficient decrease of the function values. To this aim, besides the directional derivatives, we must take into account the magnitudes of the feasible steplengths. Denoting by (β^k, d^k) a solution of (27), we have that β^k is the minimum between the maximum feasible steplength along d^k and the given positive scalar β_u. Then, our strategy at each iteration k is based on the following sequential steps:

– given the feasible current point x^k satisfying (15), among the free indexes $R(x^k) \cap S(x^k)$, we identify an index $i(k)$ corresponding to a "sufficiently free" component, that is such that

$$\min\{x^k_{i(k)} - l_{i(k)}, u_{i(k)} - x^k_{i(k)}\} \geq \bar{\eta} > 0,$$

where $\bar{\eta}$ is a prefixed sufficiently small positive scalar; in the case that

$$\min\{x^k_i - l_i, u_i - x^k_i\} < \bar{\eta}, \quad \forall i \in \{1, \dots, n\},$$

we select an index $i(k) \in \{1, \dots, n\}$ such that

$$\min\{x^k_{i(k)} - l_{i(k)}, u_{i(k)} - x^k_{i(k)}\} \geq \min\{x^k_i - l_i, u_i - x^k_i\}, \quad \forall i \in \{1, \dots, n\},$$

that is $i(k)$ corresponds to the "most free" component;
– given the subset of feasible directions $D^{i(k)}(x^k)$ defined as in (28), we determine a solution (β^k, d^k) of (27), which should produce a direction of "sufficient" decrease of the objective function;
– we compute a feasible point $\tilde{x}^{k+1}$ by means of the Armijo-type line search along the direction d^k to obtain a reference value $f^k_{\text{ref}} = f(\tilde{x}^{k+1})$;
– the current point x^k is updated to a feasible point x^{k+1} such that (15) holds and

$$f(x^{k+1}) \leq f^k_{\text{ref}}.$$

The algorithm is formally described below, where we denote by η^k the following positive scalar:

$$\eta^k = \max_{h \in R(x^k) \cap S(x^k)} \{\min\{x^k_h - l_h, u_h - x^k_h\}\}. \tag{29}$$

Algorithm 5.1 (Feasible Directions (FEDIR) Algorithm)

Data. A feasible point x^0 such that $l_h < x^0_h < u_h$ for some $h \in \{1, \dots, n\}$; $\bar{\eta} > 0$, $\beta_u > 0$.

Initialization. Set $k = 0$.

While the stopping criterion is not satisfied, execute Steps 1–7 below.

Step 1. Select $i(k) \in R(x^k) \cap S(x^k)$ such that

$$\min\{x^k_{i(k)} - l_{i(k)}, u_{i(k)} - x^k_{i(k)}\} \geq \min\{\eta^k, \bar{\eta}\}.$$

Step 2. Let $D^{i(k)}(x^k)$ be the set of feasible directions of x^k defined by (28); then, select d^k, β^k such that

$$\beta^k \nabla f(x^k)^T d^k = \min_{\beta\in R, d\in R^n} \beta \nabla f(x^k)^T d, \tag{30a}$$

$$0 \le \beta \le \beta_u, \tag{30b}$$

$$d \in D^{i(k)}(x^k), \tag{30c}$$

$$x^k + \beta d \in \mathcal{F}. \tag{30d}$$

Step 3. If $\nabla f(x^k)^T d^k \geq 0$ then stop; otherwise, compute the stepsize λ^k using Algorithm ALS(x^k, d^k, β^k).
Step 4. Set $\tilde{x}^{k+1} = x^k + \lambda^k d^k$ and $f_{\text{ref}}^k = f(\tilde{x}^{k+1})$.
Step 5. Find $x^{k+1} \in \mathcal{F}$ such that (15) holds and $f(x^{k+1}) \leq f_{\text{ref}}^k$.
Step 6. Set $k = k + 1$.
Step 7. Return $x^* = x^k$.

The role of Step 1 and Step 2 is that of finding a descent direction $d^k \in D^{i(k)}(x^k)$ whose two nonzero elements in position $i(k)$, $j(k)$, identify the two components of x^k that will be changed, by Step 3 and Step 4, to determine the reference point $\tilde{x}^{k+1}$. In other words, the pair $(i(k), j(k))$ identifies the working set W^k which leads, in a traditional decomposition scheme, to subproblem (2). Concerning Step 3, in the case that x^k is not a stationary point, that is $\nabla f(x^k)^T d^k < 0$, we have that Algorithm ALS performs an inexact one dimensional constrained minimization along the search direction d^k, thus computing a suitable stepsize λ^k. Note that Proposition 3.5 guarantees that d^k is such that $\nabla f(x^k)^T d^k < 0$ whenever x^k is not a stationary point. Therefore, as d^k has only two nonzero components of the form either $1/a_{i(k)}$ and $-1/a_{j(k)}$, or $-1/a_{i(k)}$ and $1/a_{j(k)}$, we have $\|d^k\| \leq \sqrt{2}/\{\min_{1\le i\le n}\{|a_i|\}$, it follows that Assumption 4.1 is satisfied and hence the properties of Algorithm ALS (stated in Proposition 4.1) hold. We remark that in the quadratic case (either convex or non convex) Step 3 can be simplified. Indeed, the optimal feasible stepsize λ^* along d^k can be analytically determined. In particular, denoting by Q the Hessian matrix of the objective function, we have

$$\lambda^* = \begin{cases} \min\{-\nabla f(x^k)^T d^k/(d^k)^T Q d^k, \beta_{\mathcal{F}}^k\}, & \text{if } (d^k)^T Q d^k > 0, \\ \beta_{\mathcal{F}}^k, & \text{otherwise.} \end{cases}$$

Then, at Step 4 we can set $\tilde{x}^{k+1} = x^k + \lambda^\star d^k$ since the convergence properties of the Armijo rule are preserved (see, e.g., Ref. [20]). Finally, we point out that at Step 5 we have the possibility of defining the updated point x^{k+1} in any way provided that (15) holds and $f(x^{k+1}) \leq f_{\text{ref}}^k$. Therefore, we can employ, in principle, a working set of arbitrary dimension and arbitrary elements to compute the new point x^{k+1}. This is an important aspect both from a theoretical and a practical point of view.

6 Convergence Analysis

Before analyzing the convergence properties of the decomposition algorithm model, we need to introduce the following assumption, which guarantees that every point produced by the algorithm has at least one "free" component and, hence, that condition (15) holds.

We denote by V the set of feasible points without "free" components, i.e. (see (4)),

$$V = \{x \in \mathcal{F} : R(x) \cap S(x) = \emptyset\},$$

and, for any feasible starting point x^0, we introduce the following level set

$$\mathcal{L}^0 = \{x \in \mathcal{F} : f(x) \le f(x^0)\}.$$

Assumption 6.1 $\mathcal{L}^0 \cap V = \emptyset$.

Note that Assumption 6.1 is automatically satisfied whenever at least one variable is unbounded, namely $l_i = -\infty$, $u_i = +\infty$ for some $i \in \{1, \dots, n\}$. In connection with SVM problems where all the variables are bounded, we will prove (see Proposition B.1) in the appendix that, under suitable conditions, Assumption 6.1 holds.

The convergence properties of the algorithm are stated in the next proposition.

Proposition 6.1 *Let $\{x^k\}$ be the sequence generated by the algorithm and let Assumption 6.1 hold. Then, every limit point of $\{x^k\}$ is a stationary point of Problem (1).*

Proof The instructions of the algorithm imply $f(x^{k+1}) \le f(x^k)$ for all $k \ge 0$, and hence the sequence of points $\{x^k\}$ belongs to the level set $\mathcal{L}^0$. Let $\bar{x}$ be any limit point of $\{x^k\}$, i.e. there exists an infinite subset $K \subseteq \{0, 1, \dots, \}$ such that $x^k \to \bar{x}$ for $k \in K, k \to \infty$. From Assumption 6.1, it follows that $\bar{x} \notin V$ and this implies that there exists a scalar $\epsilon > 0$ such that, for $k \in K$ and k sufficiently large, we have

$$\eta^k = \max_{h \in R(x^k) \cap S(x^k)} \{\min\{x_h^k - l_h, u_h - x_h^k\}\} \ge \min\{\epsilon, \bar{\eta}\}. \tag{31}$$

Now, we note that $i(k) \in \{1, \dots, n\}$, so that we can extract a further subset of K (relabelled again K) and an index $i \in \{1, \dots, n\} \cap R(x^k) \cap S(x^k)$ such that

$$D^{i(k)}(x^k) = D^i(x^k),$$

for all $k \in K$. Recalling the selection rule of index i at Step 1 and (31), we can write

$$\min\{x_i^k - l_i, u_i - x_i^k\} \ge \min\{\eta^k, \bar{\eta}\} \ge \min\{\epsilon, \bar{\eta}\}.$$

Taking limits for $k \to \infty$ and $k \in K$, we obtain

$$\min\{\bar{x}_i - l_i, u_i - \bar{x}_i\} \ge \min\{\epsilon, \bar{\eta}\} > 0,$$

from which we get that $i \in R(\bar{x}) \cap S(\bar{x})$. Then, (18) implies that

$$\text{cone}(D^i(\bar{x})) = D(\bar{x}).$$

Assume now, by contradiction, that $\bar{x}$ is not a stationary point of Problem (1). Then, by Proposition 3.5, there exists a direction $\hat{d} \in D^i(\bar{x})$ such that

$$\nabla f(\bar{x})^T \hat{d} < 0. \tag{32}$$

By definition (28), we have

$$D^i(x^k) = \{d^{r,i} \in R^n : r \in R(x^k)\} \cup \{d^{i,s} \in R^n : s \in S(x^k)\}.$$

Furthermore, from Proposition 2.3 we have $R(\bar{x}) \subseteq R(x^k)$ and $S(\bar{x}) \subseteq S(x^k)$ for $k \in K$ and k sufficiently large. Therefore, for $k \in K$ and k sufficiently large, it follows that $D^i(\bar{x}) \subseteq D^i(x^k)$ and hence that

$$\hat{d} \in D^i(x^k). \tag{33}$$

Let $\hat{\beta}^k$ be the maximum feasible stepsize along $\hat{d}$ starting from x^k. From Proposition A.1, reported in the Appendix, we have

$$\hat{\beta} \le \hat{\beta}^k \le +\infty, \tag{34}$$

for some $\hat{\beta} > 0$. Note that, from (32) and (33), it follows that the direction d^k and the steplength β^k selected at Step 2 are such that

$$\beta^k \nabla f(x^k)^T d^k \le \min\{\hat{\beta}^k, \beta_u\} \nabla f(x^k)^T \hat{d} < 0. \tag{35}$$

Furthermore, we have $\|d^k\| \le \sqrt{2}/\{\min_{1\le i\le n}\{|a_i|\}$ and hence Assumption 4.1 on $\{d^k\}$ holds. Step 5 of the algorithm implies that

$$f(x^{k+1}) \le f(\tilde{x}^{k+1}) = f(x^k + \lambda^k d^k) \le f(x^k) + \gamma\lambda^k \nabla f(x^k)^T d^k, \quad \gamma \in (0, 1/2), \tag{36}$$

where λ^k is the stepsize computed by Algorithm ALS. The sequence $\{f(x^k)\}$ is monotone decreasing and convergent to a finite value since, by the continuity of f, we have

$$\lim_{k\to\infty, k\in K} f(x^k) = f(\bar{x}).$$

Therefore, (36) implies that (25) holds. Thus, from assertion (ii) of Proposition 4.1 we obtain (26) for $k \in K$, where $\beta^k = \min\{\beta_f^k, \beta_u\}$, and β_f^k is the maximum feasible stepsize along d^k. Then, taking limits in (35) for $k \to \infty$, $k \in K$, and using (26) and (34), we obtain

$$\nabla f(\bar{x})^T \hat{d} = 0,$$

but this contradicts (32). □

7 Numerical Experiments on SVM Training Problems

In this section we present a specific realization of FEDIR algorithm for the particular class of problems arising in large-scale SVM training. We report the numerical results obtained on standard test problems. Given a training set of input-target pairs (u^i, a^i), $i = 1, \dots, n$, with $u^i \in R^m$, and $a^i \in \{-1, 1\}$, the SVM classification technique (Ref. [6]) requires the solution of the following quadratic programming problem:

$$\begin{aligned} \min \quad & f(x) = 1/2 x^T Q x - e^T x, && \text{(37a)} \\ \text{s.t.} \quad & a^T x = 0, && \text{(37b)} \\ & 0 \le x \le Ce, && \text{(37c)} \end{aligned}$$

where $x \in R^n$, Q is a $n \times n$ symmetric positive semidefinite matrix, $e \in R^n$ is the vector of all ones, $a \in \{-1, 1\}^n$ and C is a positive scalar. The generic element q_{ij} of the matrix Q is given by $a^i a^j K(u^i, u^j)$, where $K(u, z) = \phi(u)^T \phi(z)$ is the kernel function related to the nonlinear function ϕ that maps the data from the input space into the feature space. One of the most popular and efficient algorithm for SVM training is LIBSVM algorithm (Ref. [21]). This is a decomposition method where, at each iteration k, the working set W of subproblem (2) is determined by the two nonzero components of the solution of problem

$$\min_d \left\{ \nabla f(x^k)^T d : d \in D(x^k), -e \le d \le e, \left| \{i : d_i \ne 0\} \right| = 2 \right\}. \tag{38}$$

LIBSVM (version 2.71) algorithm can in turn be viewed as a special case of the SVM^{light} algorithm (Ref. [22]), which is based on a specific procedure for choosing the q elements of the working set, with q being any even number. In Sect. II of Ref. [14] it has been proved that an optimal solution of problem (38) belongs to $D_{RS}(x^k)$, so that such a solution can be found by solving the following problem:

$$\begin{aligned} \min_{d \in R^n} \quad & \nabla f(x^k)^T d, && \text{(39a)} \\ \text{s.t.} \quad & d \in D_{RS}(x^k); && \text{(39b)} \end{aligned}$$

further a solution of problem (39) is a descent direction for f at x^k. The selection of the two indices by solving problem (39) requires $O(n)$ operations. The two indices are those corresponding to the "maximal violation" of the KKT conditions (see Proposition 2.2), and hence the selection rule is usually referred as the Maximal Violation (MV) rule (Ref. [23]). We observe that, in the case of working set W of cardinality two, the optimal solution of subproblem (2) can be analytically computed (see, e.g., Ref. [24]). We remark that we are assuming that the Hessian matrix cannot be stored, and hence in computational terms the most expensive step of a general decomposition method is the evaluation of the columns of the Hessian matrix, corresponding to the indices in the working set W, not stored in memory. Actually, these columns are needed for updating the gradient, given in the quadratic case by

$$\nabla f(x^{k+1}) = \nabla f(x^k) + \sum_{i=1}^n Q_i (x^{k+1} - x^k)_i,$$

where Q_i is the i-th column of Q. To reduce the computational time a commonly adopted strategy is based on the use of a *caching technique* that allocates some memory space (the cache) to store the recently used columns of Q thus avoiding in some cases the recomputation of these columns. Algorithms based on the Maximal Violation (MV) rule have theoretical convergence properties and are efficient from a computational point of view, but they are not designed to fully exploit the information on the matrix stored in the cache. We define here a specific realization of FEDIR algorithm for SVM training, called FEDIR-SVM, where a caching strategy can be advantageously exploited. We recall that at Step 4 of FEDIR algorithm a reference value $f^k_{\text{ref}} = f(\tilde{x}^{k+1})$ is determined by performing an exact line search along d^k. Different realizations can be derived from FEDIR scheme by specifying the rule to select, at Step 5, the new iterate x^{k+1} for which it is required that (15) holds and that $f(x^{k+1}) \le f^k_{\text{ref}}$. In our proposed approach, besides the reference point $\tilde{x}^{k+1}$, two tentative points, x_a^{k+1} and x_b^{k+1}, are analytically generated: the point x_a^{k+1} is determined according to the MV strategy, the point x_b^{k+1} is defined to try to exploit the information in the cache memory thus reducing the computational time. Concerning the vector x_a^{k+1}, it is obtained as

$$x_a^{k+1} = x^k + \lambda_a^k d_a^k,$$

where d_a^k is the solution of (39) and

$$\lambda_a^k = \arg \min_{\lambda \in [0, \beta_a^k]} f(x^k + \lambda d_a^k),$$

where β_a^k is the maximum feasible steplength along d_a^k (note that we have $\beta_a^k \le C$ since C is the box size and the nonzero components of the search direction d_a^k belong to $\{-1, 1\}$). Hence, as d_a^k is a descent direction for f at x^k, it follows that $f(x_a^{k+1}) < f(x^k)$. As regards the definition of the point x_b^{k+1}, let $I_C \subset \{1, \dots, n\}$ be the set of indices (excluding i_a and j_a) of the columns of Q stored in the cache memory and let

$$R_c(x^k) = R(x^k) \cap I_C^k, \qquad S_c(x^k) = S(x^k) \cap I_C^k.$$

We apply the most violating strategy to the indices corresponding to the columns stored in the cache memory so that we define, in the case that both the sets $R_c(x^k)$ and $S_c(x^k)$ are not empty, the direction d_b^k, with two nonzero components identified by the indices i_b and j_b, as the solution of

$$\min_{d \in R^n} \quad \nabla f(x^k)^T d, \tag{40a}$$

$$\text{s.t.} \quad d \in D_{R_c S_c}(x^k), \tag{40b}$$

where

$$D_{R_c S_c}(x) = \bigcup_{\substack{i \in R_c(x) \\ j \in S_c(x) \\ i \ne j}} d^{i,j}.$$

Note that it is not guaranteed that d_b^k is a descent direction for f at x^k. In the case that $\nabla f(x^k)^T d_b^k < 0$, we find the value

$$\lambda_b^k = \arg \min_{\lambda \in [0, \beta_b^k]} f(x^k + \lambda d_b^k),$$

where β_b^k is the maximum feasible steplength along d_b^k (we have $\beta_b^k \le C$ for the same reasons explained above), and we set

$$x_b^{k+1} = x^k + \lambda_a^k d_a^k + \lambda_b^k d_b^k. \tag{41}$$

The vector x_b^{k+1} can be seen as the result of a simple procedure for inexactly solving the four-dimensional subproblem (2) with working set $W = \{i_a, j_a, i_b, j_b\}$. We observe that the computation of d_b^k and λ_b^k does not involve any additional gradient evaluation since they are computed starting from x^k, where the gradient $\nabla f(x^k)$, necessary to check the descent condition of d_b^k and to determine the optimal stepsize λ_b^k, is available (the reasonable choice of starting from x_a^{k+1} should imply an additional cost due to the evaluation of $\nabla f(x_a^{k+1})$). Summarizing, at Step 5 of FEDIR scheme we define x^{k+1} as the point whose function value corresponds to the minimum either between f_{ref}^k and $f(x_a^{k+1})$, or between f_{ref}^k, $f(x_a^{k+1})$ and $f(x_b^{k+1})$ whenever the point x_b^{k+1} has been generated. At each iteration either two or four variables are updated. In any case, the updated point is analytically determined. Note that the most expensive task is the evaluation of the columns of the Hessian matrix, corresponding to the modified variables, not stored in memory. Thanks to the information stored in the cache memory, at most two columns of the Hessian matrix must be possibly recomputed. In order to make some fair computational comparison between LIBSVM and FEDIR-SVM, this latter has been implemented by suitably modifying the available code (written in C++) of LIBSVM. In our computational experiments the classification test problems described below have been used. All the problems have been taken from LIBSVM Database (Ref. [25]), except for the random problem, and n, m denote the number of training pairs and the dimension of the input space respectively.

- a4a: $n = 4781$, $m = 123$;
- mushrooms: $n = 8124$, $m = 112$;
- w5a: $n = 9888$, $m = 300$;
- rcv1_train.binary: $n = 20242$, $m = 47236$;
- ijcnn1: $n = 49990$, $m = 22$;
- random: $n = 50000$, $m = 20$; it has been constructed starting from two linearly separable sets of points by changing the classification of a certain number (equal to 1% of the overall points) of randomly chosen observations (Refs. [7, 15]).

The experiments have been performed using in (37) the Gaussian kernel $K(u, z) = e^{-\gamma \|u-z\|^2}$, with $\gamma = 1$, i.e., the generic element $q_{i,j}$ of the Hessian matrix Q is $a^i a^j e^{-\|u^i - u^j\|^2}$, and setting the upper bound C of the box constraints equal to 5. The initial point x^0 in FEDIR-SVM has been determined starting from the feasible point $x = 0$ and applying the MV strategy, which corresponds to performing the first iterate of LIBSVM. Furthermore, we have set in FEDIR-SVM $\bar{\eta} = 10^{-6}$, $\beta_u = C = 5$,

Table 1 Numerical results

Problem (dimension n)	Algorithm	n_i	K_{ev}	f^*	CPU
a4a	FEDIR-SVM	5538	10331	−2358.74	34
($n = 4781$)	LIBSVM	6599	10714	−2358.74	34
mushrooms	FEDIR-SVM	12193	24386	−1072.91	157
($n = 8124$)	LIBSVM	16083	32152	−1072.91	203
w5a	FEDIR-SVM	9141	17272	−789.39	102
($n = 9888$)	LIBSVM	10217	19400	−789.39	113
rcv1_train.binary	FEDIR-SVM	10460	20436	−2498.03	817
($n = 20242$)	LIBSVM	13570	24358	−2498.03	971
ijcnn1	FEDIR-SVM	12666	13128	−12986.54	354
($n = 49990$)	LIBSVM	26741	20196	−12986.54	544
random	FEDIR-SVM	26435	52804	−7673.07	1430
($n = 50000$)	LIBSVM	40559	80822	−7673.07	2230

and we have used the same stopping criterion adopted by LIBSVM (Ref. [21]). All the experiments have been carried out on a 2.60 GHz Pentium 4 with 512 megabytes of RAM and cache size of 40 MB. The results are shown in Table 1, where we report the number of iterations (n_i), the number of column evaluations of the matrix Q (K_{ev}), the attained function value (f^*), and the required CPU time (CPU) expressed in seconds. From the results reported in Table 1 we can observe that the number of iterations and the number of matrix column evaluations required by FEDIR-SVM are always lower than those required by LIBSVM. Concerning the comparison in terms of number of iterations, we have that in many cases FEDIR-SVM accepts as new iterate the point x_b^{k+1} (see (41)) thus updating four variables, while LIBSVM always updates two variables, and this motivates the better performances of FEDIR-SVM. As regards the better behaviour of FEDIR-SVM in terms of number of matrix column evaluations, from one hand it depends on the lower number of iterations, from the other one it depends on the fact that FEDIR-SVM is designed to exploit as much as possible the caching strategy. As consequence, FEDIR-SVM outperforms LIBSVM in terms of CPU time in all the runs except for problem *a4a*. We can note that the difference of the performances of the two algorithms becomes significant as the dimension of the problem increases. Thus, the computational experiments performed on a class of SVM large-scale quadratic problems show the effectiveness of the implemented decomposition scheme derived from the general algorithm model proposed in this work.

Acknowledgements The authors wish to thank the anonymous reviewer for helpful comments and constructive suggestions.

Appendix A: Technical Result

We state here a theoretical result valid for polyhedral sets. This result is used in the convergence proof of the general algorithm. More specifically, we consider a polyhedron

$$\Gamma = \{x \in R^n : Ax \leq b\},$$

where A is a $m \times n$ matrix and $b \in R^m$. Given $x \in \Gamma$, we denote by $I(x)$ the set of active constraints at x, that is,

$$I(x) = \{i \in \{1, \dots, m\} : a_i^T x = b_i\},$$

where a_i^T denotes row i of A. The set of feasible directions $D(x)$ at $x \in \Gamma$ is

$$D(x) = \{d \in R^n : a_i^T d \leq 0 \ \ \forall i \in I(x)\}.$$

Given $d \in D(x)$, we denote by $\mathcal{H} = \{i \in \{1, \dots, m\} : a_i^T d > 0\}$. The *maximum feasible stepsize* along d is

$$\beta = \begin{cases} +\infty, & \text{if } \mathcal{H} = \emptyset, \\ \min_{i \in \mathcal{H}}\{(b_i - a_i^T x)/a_i^T d\}, & \text{otherwise.} \end{cases} \tag{42}$$

Proposition A.1 *Let $A \in R^{m \times n}$ and $b \in R^m$. Let $\{x^k\}$ be a sequence of points such that*

$$Ax^k \leq b, \tag{43}$$

for all k. Assume further that

$$\lim_{k \to \infty} x^k = \bar{x}. \tag{44}$$

Then, given any direction $\bar{d} \in D(\bar{x})$, there exists a scalar $\hat{\beta} > 0$ such that, for sufficiently large values of k, we have

$$A(x^k + \beta \bar{d}) \leq b, \quad \forall \beta \in [0, \hat{\beta}]. \tag{45}$$

Proof Assume first that $\mathcal{H} = \emptyset$. Then, recalling (42), it follows that (45) holds for any $\hat{\beta} > 0$. Now, suppose that $\mathcal{H} \neq \emptyset$ and let $\bar{\beta}$ be the maximum feasible stepsize along $\bar{d}$ starting from $\bar{x}$, i.e.,

$$\bar{\beta} = \min_{i \in \mathcal{H}}\{(b_i - a_i^T \bar{x})/a_i^T \bar{d}\} > 0$$

(the inequality holds since $\bar{d} \in D(\bar{x})$). Then, we have

$$a_i^T(\bar{x} + \beta \bar{d}) < b_i, \quad \forall i \in \mathcal{H}, \forall \beta \in [0, \bar{\beta}/2],$$

and hence there exists a scalar $\epsilon > 0$ such that

$$a_i^T(\bar{x} + \beta \bar{d}) \leq b_i - \epsilon, \quad \forall i \in \mathcal{H}, \forall \beta \in [0, \bar{\beta}/2]. \tag{46}$$

Moreover, we have

$$a_j^T(\bar{x}+\beta\bar{d}) \le b_j, \quad \forall j \notin \mathcal{H}, \forall \beta \in [0,+\infty). \tag{47}$$

Recalling (44), for k sufficiently large, we can write

$$|a_i^T x^k - a_i^T \bar{x}| \le \epsilon/2, \quad \forall i \in \{1,\dots,m\}. \tag{48}$$

Finally, using (46) and (48), for k sufficiently large and $\forall i \in \mathcal{H}$, we have

$$a_i^T(x^k+\beta\bar{d}) = a_i^T(x^k+\bar{x}-\bar{x}+\beta\bar{d}) \le b_i - \epsilon + \epsilon/2 = b_i - \epsilon/2, \quad \forall \beta \in [0,\bar{\beta}/2] \tag{49}$$

so that, using (47), (45) is proved with $\hat{\beta} = \bar{\beta}/2$. □

Appendix B: SVM Problem and Assumption 6.1

The next proposition identifies a class of SVM training problems which satisfy Assumption 6.1.

Proposition B.1 *Consider the convex quadratic problem* (37), *and let x^0 be a feasible point such that $f(x^0) < 0$. Suppose that*

$$\lambda_{\min}(Q) - 2/C \ge 0, \tag{50}$$

where $\lambda_{\min}(Q)$ denotes the minimum eigenvalue of Q. Then, Assumption 6.1 holds.

Proof In order to prove the thesis, it suffices to show that, given any point x belonging to the set

$$V = \{x \in \mathcal{F} : R(x) \cap S(x) = \emptyset\},$$

we have $f(x) \ge 0$. For any $x \in V$, we have

$$\|x\|^2 = \sum_{i:x_i=C} C^2, \qquad e^T x = \sum_{i:x_i=C} C,$$

and we can write

$$f(x) \ge 1/2\lambda_{\min}(Q)\|x\|^2 - e^T x = C \sum_{i:x_i=C} (1/2\lambda_{\min}(Q)C - 1) \ge 0,$$

where the last inequality follows from (50). □

Finally we remark that the requirement that the initial feasible point x^0 is such that $f(x^0) < 0$ can be easily satisfied in the case of SVM problems. In fact, for such problems, the point $\tilde{x} = 0$ is feasible and, whenever this point is not a stationary point, it is possible to find a feasible descent direction (see Propositions 3.2 and 3.3). Therefore, a point x^0 such that $f(x^0) < 0$ can be found by starting from $\tilde{x}$ and by performing a line search along this direction.

References

1. Barr, R.O., Gilbert, E.G.: Some efficient algorithms for a class of abstract optimization problems arising in optimal control. IEEE Trans. Autom. Control **14**, 640–652 (1969)
2. Correa, J.R., Schulz, A.S., Stier Moses, N.E.: Selfish routing in capacitated networks. Math. Oper. Res. 961–976 (2004)
3. Kao, C., Lee, L.-F., Pitt, M.M.: Simulated maximum likelihood estimation of the linear expenditure system with binding non-negativity constraints. Ann. Econ. Finance **2**, 203–223 (2001)
4. Meyer, R.R.: Multipoint methods for separable nonlinear networks. Math. Program. Study **22**, 185–205 (1985)
5. Bomze, I.M.: Evolution towards the maximum clique. J. Glob. Optim. **10**, 143–164 (1997)
6. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
7. Ferris, M.C., Munson, T.S.: Interior-point methods for massive support vector machines. SIAM J. Optim. **13**, 783–804 (2003)
8. Ferris, M.C., Munson, T.S.: Semismooth support vector machines. Math. Program. **101**, 185–204 (2004)
9. Lee, Y.-J., Mangasarian, O.L.: SSVM: a smooth support vector machine for classification. Comput. Optim. Appl. **20**, 5–22 (2001)
10. Mangasarian, O.L., Thompson, M.E.: Massive data classification via unconstrained support vector machines. J. Optim. Theory Appl. **131**(3), 315–325 (2006)
11. Mangasarian, O.L., Musicant, D.R.: Active set support vector machine classification. In: Lee, T.K., Dietter, T.G. (eds.) Neural Information Processing Systems 2000 (NIPS 2000), pp. 577–583. MIT Press, Cambridge (2001)
12. Scheinberg, K.: An efficient implementation of an active set method for SVMs. J. Mach. Learn. Res. **7**, 2237–2257 (2006)
13. Dai, Yu.-H., Fletcher, R.: New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. Math. Program. **106**, 403–421 (2006)
14. Lin, C.-J.: On the convergence of the decomposition method for support vector machines. IEEE Trans. Neural Netw. **12**, 1288–1298 (2001)
15. Lucidi, S., Palagi, L., Risi, A., Sciandrone, M.: A convergent decomposition algorithm for support vector machines. Comput. Optim. Appl. **38**, 217–234 (2007)
16. Mangasarian, O.L., Musicant, D.R.: Successive overrelaxation for support vector machines. IEEE Trans. Neural Netw. **10**, 1032–1037 (1999)
17. Serafini, T., Zanni, L.: On the working set selection in gradient projection-based decomposition techniques for support vector machines. Optim. Methods Softw. **20**, 583–596 (2005)
18. Chang, C.-C., Hsu, C.-W., Lin, C.-J.: The analysis of decomposition methods for support vector machines. IEEE Trans. Neural Netw. **11**, 1003–1008 (2000)
19. Palagi, L., Sciandrone, M.: On the convergence of a modified version of SVM^{light} algorithm. Optim. Methods Softw. **20**, 311–328 (2005)
20. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, New York (1999)
21. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm (2001)
22. Joachims, T.: Making large scale SVM learning practical. In: Schölkopf, C.B.B., Smola, A. (eds.) Advances in Kernel Methods—Support Vector Learning. MIT Press, Cambridge (1998)
23. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Trans. Neural Netw. **11**, 124–136 (2000)
24. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)
25. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. LIBSVM dataset: classification, regression and multi-label