On the Convergence of the Decomposition Method for Support Vector Machines

Chih-Jen Lin

Department of Computer Science and

Information Engineering

National Taiwan University

Taipei 106, Taiwan

cjlin@csie.ntu.edu.tw

Abstract

The decomposition method is currently one of the major methods for solving support vector machines (SVM). Its convergence properties have not been fully understood. The general asymptotic convergence was first proposed by Chang et al. [2]. However, their working set selection does not coincide with existing implementation. A later breakthrough by Keerthi and Gilbert [10] proved the convergence for practical cases while the size of the working set is restricted to two. In this paper, we prove the convergence of the algorithm used by the software SVM^{light} [9] and other later implementation. The size of the working set can be any even number. Extensions to other SVM formulations are also discussed.

Keywords

Support vector machines, decomposition methods, classification.

I. Introduction

The support vector machine (SVM) is a new and promising technique for classification. Surveys of SVM are, for example, Vapnik [25], [26] and Schölkopf et al. [21]. Given training vectors $x_i \in R^n, i = 1, \ldots, l$, in two classes, and a vector $y \in R^l$ such that $y_i \in \{1, -1\}$, the support vector technique requires the solution of the following optimization problem:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\
& 0 \le \alpha_i \le C, i = 1, \ldots, l, \qquad (1)\\
& y^T\alpha = 0,
\end{aligned}
$$

where e is the vector of all ones, C is the upper bound of all variables, and Q is an l by l positive semidefinite matrix. Training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function ϕ and $Q_{ij} \equiv y_i y_j K(x_i, x_j)$ where $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the kernel.

The difficulty of solving (1) is the density of Q because Q_{ij} is in general not zero. In this case, Q becomes a fully dense matrix so a prohibitive amount of memory is required to store the matrix. Thus traditional optimization algorithms such as Newton, Quasi Newton, etc., cannot be directly applied. Several authors (for example, Osuna et al. [16], Joachims [9], Platt [17], and Saunders et al. [20]) have proposed decomposition methods to conquer this difficulty. The basic concept of this method is as follows:

Algorithm I.1 (Decomposition method)

1. Given a number $q \leq l$ as the size of the working set. Find α^1 as the initial solution. Set $k = 1$.

2. If α^k is an optimal solution of (1), stop. Otherwise, find a working set $B \subset \{1, \ldots, l\}$ whose size is q. Define $N \equiv \{1, \ldots, l\} \backslash B$ and α_B^k and α_N^k to be sub-vectors of α^k corresponding to B and N, respectively.

3. Solve the following sub-problem with the variable α_B:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\alpha_B^T Q_{BB}\alpha_B - (e_B - Q_{BN}\alpha_N^k)^T \alpha_B \\
& 0 \leq (\alpha_B)_i \leq C, i = 1, \ldots, q, \\
& y_B^T \alpha_B = -y_N^T \alpha_N^k,
\end{aligned}
\tag{2}
$$

where $\left[\begin{smallmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{smallmatrix}\right]$ is a permutation of the matrix Q.

4. Set α_B^{k+1} to be the optimal solution of (2) and $\alpha_N^{k+1} \equiv \alpha_N^k$. Set $k \leftarrow k+1$ and goto Step 2.

The basic idea of the decomposition method is that in each iteration, the indices $\{1, \ldots, l\}$ of the training set are separated to two sets B and N, where B is the working set and $N = \{1, \ldots, l\} \backslash B$. The vector α_N is fixed so the objective value becomes $\frac{1}{2}\alpha_B^T Q_{BB}\alpha_B - (e_B - Q_{BN}\alpha_N)^T \alpha_B + \frac{1}{2}\alpha_N^T Q_{NN}\alpha_N - e_N^T \alpha_N$. Then a sub-problem with the variable α_B, i.e. (2), is solved. Note that B is updated in each iteration. To simplify the notation, we simply use B instead of B^k.

An important issue of the decomposition method is to select the working set B in each iteration (Step 2 of Algorithm I.1). Among existing methods, Osuna et al. [16], and Saunders et al. [20] find the working set by choosing elements which violate the Karush-Kuhn-Tucker (KKT) condition. Platt's Sequential Minimal Optimization (SMO) [17] restricts the size of the working set to be two. The advantage is that (2) becomes a small problem so no optimization software is required in practice. His working selection includes some heuristics. A systematic way is proposed by Joachims [9] where he restricts q to be an *even* number. In his software SVM^{light*}, the following problem with the variable d is solved:

$$\begin{aligned} \min \quad & \nabla f(\alpha^k)^T d \\ & y^T d = 0, \ -1 \le d_i \le 1, i = 1, \dots, l, && \text{(3a)} \\ & d_i \ge 0, \ \text{if } (\alpha^k)_i = 0, \qquad d_i \le 0, \ \text{if } (\alpha^k)_i = C, && \text{(3b)} \\ & |\{d_i \mid d_i \neq 0\}| \le q, && \text{(3c)} \end{aligned}$$

where we represent $f(\alpha) \equiv \frac{1}{2}\alpha^T Q \alpha - e^T \alpha$, α^k is the solution at the kth iteration, and $\nabla f(\alpha^k)$ is the gradient of $f(\alpha)$ at α^k. Note that $|\{d_i \mid d_i \neq 0\}|$ means the number of components of d which are not zero. The constraint (3c) implies that a direction d involving only q variables is obtained. Then components of α^k with non-zero d_i are included in the working set B which is used to construct the sub-problem (2). Note that d is only used for identifying B but not as a search direction. In Joachims' original paper, $|\{d_i \mid d_i \neq 0\}| = q$ instead of (3c) was used. Thus practically the decomposition method always picks q elements in each iteration. It was first pointed out in [2] that in theory q nonzero elements may not be always available so an inequality (3c) was proposed.

Joachims [9] used the following procedure to solve (3):

Algorithm I.2 (**SVM^{light}'s working set selection**)

1. Sort $y_i \nabla f(\alpha^k)_i$ in the decreasing order.

2. From the top of the sorted list sequentially set $d_i = -y_i$ if $0 < \alpha_i^k < C$ or (3b) is satisfied. If $d_i = -y_i$ violates (3b), set $d_i = 0$ and bypass it. From the bottom of the list sequentially set $d_i = y_i$ if $0 < \alpha_i^k < C$ or (3b) is satisfied. If $d_i = y_i$ violates (3b), set $d_i = 0$ and bypass it. The assignment of $d_i = -y_i$ and y_i is done symmetrically until either

$^*SVM^{light}$ is available at
`http://www-ai.cs.uni-dortmund.de/FORSCHUNG/VERFAHREN/SVM_LIGHT/svm_light.eng.html`

(a) $q/2$ elements of d are assigned to be $-y_i$ from the top and $q/2$ elements of d are assigned to be y_i from the bottom; or

(b) we cannot find $d_i = -y_i$ from the top and $d_i = y_i$ from the bottom at the same time.

3. Elements of d not considered yet are assigned to be zeros.

Algorithm I.2 will be discussed in more detail later. We mention the working set selection here because it is strongly related to the main topic of this paper: the convergence of the decomposition method.

As the decomposition method finds an optimal solution of a sub-problem (2), the strict decrease of the objective function holds. However, this does not imply that the sequence $\{\alpha^k\}$ converges to an optimal solution of (1). In fact the convergence issue is not easy and has not been fully understood yet.

The first work on the convergence of the decomposition method is by Chang et al. [2]. They proved the convergence of a more generalized algorithm. However, their working set selection is by a different problem:

$$\begin{aligned} \min \quad & \nabla f(\alpha^k)^T d \\ & 0 \leq \alpha_i^k + d_i \leq C, i = 1, \ldots, l, \\ & y^T d = 0, \\ & |\{d_i \mid d_i \neq 0\}| \leq q. \end{aligned} \tag{4}$$

The main shortcoming is that (4) may not be useful in practice. Unlike Algorithm I.2 for (3), we have not known any comparable method for (4). Note that Algorithm I.2 takes at most $O(l \ln l)$ operations that is acceptable for practical implementation.

Then an important progress is by Keerthi and Gilbert [10] where they proved the convergence of the decomposition method with $q = 2$. In [11] the authors showed that the original SMO may not converge so some modifications and improvements were added to SMO. Then [10] intended to prove the convergence of a generalized SMO algorithm. Incidently (3) with $q = 2$ is a special case of the working set selection proposed in [11] (i.e. modification 2 of SMO in that paper). Thus their proof has covered some existing practical implementation.

Up to now the only available implementation using $q > 2$ with convergence proofs is discussed in [8]. Instead of using the standard formulation, [8] solves

$$\begin{aligned} \min \quad & \frac{1}{2}\alpha^T(Q + yy^T)\alpha - e^T\alpha \\ & 0 \le \alpha_i \le C, \qquad i = 1, \ldots, l. \end{aligned} \tag{5}$$

This formulation was proposed and studied by, for example, Mangasarian and Musicant [14], and Friess et al. [7]. (5) is a bound-constrained problem so the working set selection is by the following problem:

$$\begin{aligned} \min \quad & \nabla \bar{f}(\alpha^k)^T d \\ & 0 \le \alpha_i^k + d_i \le C, i = 1, \ldots, l, \\ & |\{d_i \mid d_i \ne 0\}| \le q, \end{aligned} \tag{6}$$

where $\bar{f}(\alpha) \equiv 1/2\alpha^T(Q + yy^T)\alpha - e^T\alpha$. The convergence follows from the framework in [2]. An important fact is that because of the simpler constraints, (6) can be solved as efficiently as solving (3). To be more precise, the complexity to solve (6) is similar to Algorithm I.2.

However, the use of (5) lacks enough theoretical support on generalization properties. We may worry that by removing the linear constraint and adding $1/2(y^T\alpha)^2$ to the objective function, the generalization property is not as good as solving (1). In addition, as more available software follow the implementation of SVM^{light} using (3) (e.g. [5], [19]), the need to prove the convergence with $q > 2$ becomes more emergent. In this paper we will show that Algorithm I.1 using (3) for the working set selection converges.

Next we discuss some possible obstacles while attempting to prove the convergence. In particular, we think the decomposition method of SVM^{light} has two major problems:

1. In each iteration, the decomposition method works only on a subset of variables. Popular optimization methods such as Newton or Quasi Newton consider all variables together in each iteration. In fact if q is small, in each iteration only few coordinates of the variable are updated. Hence the algorithm is like the "coordinate search" or "method of alternating variables" in optimization literature. It has been shown by Powell [18] that such methods may not always converge. The work in [2] focused on handling this difficulty and

a technique to construct a relationship between (4) and the following problem is utilized:

$$\begin{aligned} \min \quad & \nabla f(\alpha^k)^T d \\ & 0 \leq \alpha_i^k + d_i \leq C, i = 1, \dots, l, \\ & y^T d = 0. \end{aligned} \tag{7}$$

2. SVM^{light} uses (3) for the working set selection. Problem (3) follows from the method of feasible directions by Zoutendijk [28]. The original feasible-direction method of Zoutendijk is to consider (3) without restricting the number of nonzero elements:

$$\begin{aligned} \min \quad & \nabla f(\alpha^k)^T d \\ & y^T d = 0, -1 \leq d_i \leq 1, i = 1, \dots, l, \\ & d_i \geq 0, \text{ if } \alpha_i^k = 0, \qquad d_i \leq 0, \text{ if } \alpha_i^k = C. \end{aligned} \tag{8}$$

The difficulty arises because the convergence of Zoutendijk's method is not generally guaranteed. The main reason is that $\alpha^k + d$ may not be a feasible point of (1) so the map of search directions is not closed. An example showing that Zoutendijk's algorithm may not converge is by Wolfe [27] and more discussions are in [1]. This explains why in [2], (6) instead of (3) is considered because (6) guarantees the feasibility of $\alpha^k + d$.
However, the original Zoutendijk's method directly uses d as the search direction for the optimization algorithm. That is, a step size λ is decided and $\alpha^k + \lambda d$ becomes the next iterate α^{k+1}. This is different from the role of (3) here as d is used only for selecting the working set. Furthermore, in each iteration an exact solution of the sub-problem (2) is obtained. This seems to be a nice property which the original Zoutendijk's method lacks of. In [2], such a property was not used as they considered a more general algorithm. For the proof in this paper, we will see that it plays an important role.

The above discussion reveals that the working set selection problem (3) should be deeply investigated. In Section II, we analyze (3) and its solution procedure: Algorithm I.2. Section III is the main convergence proof. Extensions of the proof to other SVM formulations such as regression and one-class SVM are in Section IV. We make conclusions and discussions in Section V.

II. More Analysis on the Working Set Selection

Though in [9], Joachims has proposed Algorithm I.2 to solve (3), up to now there is no rigorous discussion to justify the use of this algorithm. For example, if the case 2(b) of Algorithm I.2 is encountered first, it is not clear what the practical situation looks like. In this section, we will discuss the details of Algorithm I.2 and demonstrate that it really solves (3).

First we give a simple assumption:

Assumption II.1 *$C > 0$.*

If $C = 0$, the only feasible solution of (1) $\alpha_i = 0, i = 1, \dots, l$. In addition, the constraints of (8) implies $d_i = 0$. If Algorithm I.2 is used without Assumption II.1, for $0 = \alpha_i^k = C$ we may end up with $d_i = y_i(-y_i) \neq 0$ which is not a feasible solution of (8). The assumption looks trivial but in our mind we consider the general situation $l_i \leq \alpha_i \leq u_i$, where l_i and u_i are lower and upper bounds, respectively. If $l_i = u_i$, then we can remove the ith variable from the original problem easily.

The following theorem shows how Algorithm I.2 solves (8).

Theorem II.2 *If the condition 2(a) of Algorithm I.2 is not activated (or q is selected large enough), the algorithm will finally stop at i_t (from the top) and i_b (from the bottom) and one of the following will happen:*
1. $y_{i_t}\nabla f(\alpha^k)_{i_t}$ is next to $y_{i_b}\nabla f(\alpha^k)_{i_b}$ in the sorted list.
2. There is one element $y_i\nabla f(\alpha^k)_i$ between $y_{i_t}\nabla f(\alpha^k)_{i_t}$ and $y_{i_b}\nabla f(\alpha^k)_{i_b}$ with $0 < \alpha_i^k < C$.
In addition, when the algorithm stops, d is an optimal solution of problem (8).

Proof: When Algorithm I.2 stops at i_t, if the next index in the sorted list of $y_i\nabla f(\alpha^k)_i, i = 1, \dots, l$ is $\bar{i}_t$, there are three possible situations:

$$\begin{aligned} &0 < \alpha_{\bar{i}_t}^k < C, \text{ or} \\ &\alpha_{\bar{i}_t}^k = 0, y_{\bar{i}_t} = -1, \text{ or} \\ &\alpha_{\bar{i}_t}^k = C, y_{\bar{i}_t} = 1. \end{aligned} \tag{9}$$

Otherwise, we can move down by assigning $d_{\bar{i}_t} = 0$. Then consider going up from i_b, if the next $\bar{i}_b$ is not $\bar{i}_t$ or i_t, it can not satisfy $0 < \alpha_{\bar{i}_b}^k < C$, or $\alpha_{\bar{i}_b}^k = 0, y_{\bar{i}_b} = 1$, or

$\alpha^k_{\bar{i}_b} = C, y_{\bar{i}_b} = -1$. Otherwise, (9) implies that i_t can move down to $\bar{i}_t$ and then i_b could move up. Hence $\bar{i}_b$ must satisfy $\alpha^k_{\bar{i}_b} = 0, y_{\bar{i}_b} = -1$ or $\alpha^k_{\bar{i}_b} = C, y_{\bar{i}_b} = 1$. However, for this situation, i_b could move up by setting $d_{\bar{i}_b} = 0$. Hence we are sure that there is at most one element between i_t and i_b. If there is one such element $\bar{i}_t$ and $\alpha^k_{\bar{i}_t} = 0, y_i = -1$, or $\alpha^k_{\bar{i}_t} = C, y_i = 1$, i_b could move up again by assigning $d_{\bar{i}_t} = 0$. Therefore, from (9) the only possible situation is to have an element i between i_t and i_b with $0 < \alpha^k_i < C$.

Thus we have clarified the situation when the algorithm terminates. Next we will show that when the algorithm stops, the following KKT condition is satisfied so d is an optimal solution:

$$
\begin{gathered}
\nabla f(\alpha^k) = -by + \lambda_i - \xi_i, \qquad (10)\\
y^T d = 0,\\
\lambda_i(d_i + 1) = 0, \text{ if } 0 < \alpha^k_i \leq C,\\
\lambda_i d_i = 0, \text{ if } \alpha^k_i = 0,\\
\xi_i(1 - d_i) = 0, \text{ if } 0 \leq \alpha^k_i < C,\\
\xi_i d_i = 0, \text{ if } \alpha^k_i = C,\\
\lambda_i \geq 0, \xi_i \geq 0, i = 1, \dots, l.
\end{gathered}
$$

If there is an $\bar{i}_t$ between i_t and i_b, we select b such that

$$y_{\bar{i}_t} \nabla f(\alpha^k)_{\bar{i}_t} + b = 0.$$

Otherwise, we pick b such that

$$y_i \nabla f(\alpha^k)_i + b \geq 0 \text{ for the elements before (include) } i_t \qquad (11)$$

$$y_i \nabla f(\alpha^k)_i + b \leq 0 \text{ for the elements after (include) } i_b \qquad (12)$$

Let us consider the case in (11). If $d_i = -y_i$ and $y_i = 1$, by selecting $\xi_i \equiv 0$ and $\lambda_i \equiv \nabla f(\alpha^k)_i + by_i \geq 0$, (10) is satisfied. The situation is similar for $y_i = -1$. If $d_i = 0$, there are two possibilities: $y_i = 1, \alpha^k_i = 0$ or $y_i = -1, \alpha^k_i = C$. For the first case, $\lambda_i \equiv \nabla f(\alpha^k)_i + by_i \geq 0$, $\lambda_i d_i = 0$ and $\xi \equiv 0$ satisfy (10). The argument for the second case is similar. Furthermore, the same proof can be applied for indices which satisfy

$d_i = y_i$. Thus we have shown that Algorithm I.2 obtains a KKT point. Since (8) is a linear program, a KKT point is an optimal solution. ■

After the procedure of Algorithm I.2 without activating condition 2(a) for solving (8), we assume that $i_1, \ldots, i_{m_k/2}$ are indices of elements with $d_i = -y_i$ (in the decreasing order of $\{y_i \nabla f(\alpha^k)_i\}$) and $j_1, \ldots, j_{m_k/2}$ are indices of elements with $d_i = y_i$ (in the increasing order of $\{y_i \nabla f(\alpha^k)_i\}$). Then

$$\begin{aligned} y_i \nabla f(\alpha^k)_i + b = p_i \geq 0, i = i_1, \ldots, i_{m_k/2}, \\ y_i \nabla f(\alpha^k)_i + b = n_i \leq 0, i = j_1, \ldots, j_{m_k/2}. \end{aligned}$$

We have

$$p_{i_1} \geq p_{i_2} \ldots \geq p_{i_{m_k/2}} \geq 0 \geq n_{j_{m_k/2}} \geq \ldots \geq n_{j_1}. \tag{13}$$

Since $d_i = -y_i, i = i_1, \ldots, i_{m_k/2}$ and $d_i = y_i, i = j_1, \ldots, j_{m_k/2}$,

$$\begin{aligned} \nabla f(\alpha^k)_i d_i = (b - p_i),\ i = i_1, \ldots, i_{m_k/2}, \\ \nabla f(\alpha^k)_i d_i = (-b + n_i),\ i = j_1, \ldots, j_{m_k/2}. \end{aligned}$$

Therefore, the optimal objective value of (8) is

$$\sum_{i=i_1,\ldots,i_{m_k/2}} -p_i + \sum_{i=j_1,\ldots,j_{m_k/2}} n_i.$$

Now we are ready to work on problem (3). We will show that by selecting

$$d_i \equiv \begin{cases} -y_i,\ i = i_1, \ldots, i_{\min(q,m_k)/2}, \\ y_i,\ i = j_1 \ldots, j_{\min(q,m_k)/2}, \\ 0,\ \text{otherwise}, \end{cases} \tag{14}$$

an optimal solution of (3) is obtained. When $q \geq m_k$, the solution we just obtained for (8) is a feasible solution of (3). As (3) has a smaller feasible region than (8), its objective value is not smaller. Thus d defined by (14) is an optimal solution of (3). On the other hand, if $q < m_k$, we consider the following problem:

$$\begin{aligned} \min \quad & \nabla f(\alpha^k)^T d \\ & y^T d = 0, -1 \leq d_i \leq 1, i = 1, \ldots, l, \\ & d_i \geq 0, \text{ if } \alpha_i^k = 0, \qquad d_i \leq 0, \text{ if } \alpha_i^k = C, \\ & d_i = 0, \text{ if } i \notin \bar{B}, \end{aligned} \tag{15}$$

where $\bar{B}$ is any subset of $\{1, \ldots, l\}$ containing $\bar{q}$ elements with $\bar{q} \leq q$. Now $\bar{B}$ is fixed so (15) is reduced to a form of (8) whose number of variables is $\bar{q}$. Hence the same procedure of Algorithm I.2 without Step 2(a) could be applied to solve (15). If an optimal solution is $d_i = -y_i, i = r_1, \ldots, r_{\hat{q}/2}$, $d_i = y_i, i = s_1, \ldots, s_{\hat{q}/2}$ $(\hat{q} \leq \bar{q})$, and $d_i = 0$ otherwise, then the optimal objective value of (15) is

$$(-\sum_{i=r_1,\ldots,r_{\hat{q}/2}} y_i \nabla f(\alpha^k)_i + \sum_{i=s_1,\ldots,s_{\hat{q}/2}} y_i \nabla f(\alpha^k)_i),$$

which is greater or equal to

$$(-\sum_{i=i_1,\ldots,i_{\hat{q}/2}} y_i \nabla f(\alpha^k)_i + \sum_{i=j_1,\ldots,j_{\hat{q}/2}} y_i \nabla f(\alpha^k)_i).$$

as we sort $y_i \nabla f(\alpha^k)_i$ in a decreasing order. Since $\hat{q} \leq \bar{q} \leq q$, d defined in (14) is an optimal solution of (3). The following theorem concludes the validity of Algorithm I.2 for (3):

Theorem II.3 *If q is an even positive integer, Algorithm I.2 returns an optimal solution of (3) and*

$$\frac{l}{q}(\textit{optimal objective value of (3)}) \leq \textit{optimal objective value of (8)}. \tag{16}$$

We then show the relation between the working set selection problem (3) and the original optimization problem (1):

Theorem II.4 *The optimal objective value of (3) is zero if and only if α is an optimal solution of (1).*

Proof: A basic property of Zoutendijk's method is that the optimal objective value of (8) is zero if and only if α is an optimal solution of (1) (see, for example, [1]). Since

(3) has a smaller feasible region than (8), if the optimal objective value of (3) is zero, the optimal solution of (8) is also zero. Therefore, α is an optimal solution of (1).

On the other hand, if α is an optimum of (1), with Lemma II.3, the optimal objective value of (3) is zero. ■

III. Convergence Proofs

In this section we prove the convergence of Algorithm I.1 using problem (3) for the working set selection (i.e. the algorithm used by SVM^{light}). If Algorithm I.1 stops in finite number of iterations, from Step 2, α^k is already an optimum. Hence here we consider the case where Algorithm I.1 takes infinite iterations. First we make an assumption:

Assumption III.1 *The matrix Q satisfies*

$$\min_I(\min(eig(Q_{II}))) > 0,$$

where I is any subset of $\{1,\ldots,l\}$ with $|I| \leq q$, Q_{II} is a square sub-matrix of Q, and $\min(eig(\cdot))$ *is the smallest eigenvalue of a matrix.*

If Q is positive definite, then Assumption III.1 is true. For example, if the RBF kernel $K(x_i, x_j) = e^{\|x_i - x_j\|^2}$ is used and all $x_i \neq x_j$, from [15], Q is positive definite. Since practically q is selected as a small number (≤ 100), if data are mapped into higher dimensional spaces, Q tends to be positive definite so in general Assumption III.1 holds.

The following lemma shows the sufficient decrease of $f(\alpha)$:

Lemma III.2

$$f(\alpha^{k+1}) \leq f(\alpha^k) - \frac{\sigma}{2}\|\alpha^{k+1} - \alpha^k\|^2, \tag{17}$$

where $\sigma = \min_I(\min(eig(Q_{II})))$.

Proof: Assume B is the working set at the kth iteration and $N \equiv \{1,\ldots,l\}\backslash B$. If we define $s \equiv \alpha^{k+1} - \alpha^k$, then $s_N = 0$ and

$$\begin{aligned}
& f(\alpha^{k+1}) - f(\alpha^k) \\
= \; & \frac{1}{2}s^TQs + s^TQ\alpha^k - e^Ts \\
= \; & \frac{1}{2}s_B^TQ_{BB}s_B + s_B^T(Q\alpha^k)_B - e_B^Ts_B.
\end{aligned} \tag{18}$$

That is, in the kth iteration, we solve the following problem with the variable s_B:

$$\begin{aligned} \min \quad & \frac{1}{2} s_B^T Q_{BB} s_B + s_B^T (Q\alpha^k)_B - e_B^T s_B \\ & 0 \le (\alpha^k + s)_i \le C, i \in B, \\ & y_B^T s_B = 0, \end{aligned} \tag{19}$$

which is a different representation of (2). The KKT condition of (19) shows that there is a b^{k+1} such that

$$(Q(\alpha^k + s))_i - 1 + b^{k+1} y_i = 0 \qquad \text{if } 0 < \alpha_i^k + s_i < C, i \in B, \tag{20}$$

$$(Q(\alpha^k + s))_i - 1 + b^{k+1} y_i \ge 0 \qquad \text{if } \alpha_i^k + s_i = 0, i \in B, \tag{21}$$

$$(Q(\alpha^k + s))_i - 1 + b^{k+1} y_i \le 0 \qquad \text{if } \alpha_i^k + s_i = C, i \in B. \tag{22}$$

Define $F \equiv \{i \mid 0 < \alpha_i^k + s_i < C, i \in B\}$ and $A \equiv \{i \mid \alpha_i^k + s_i = 0 \text{ or } C, i \in B\}$. We have $B = F \cup A$ and from (22),

$$\begin{aligned} (Q\alpha^k)_F &= -(Qs)_F + e_F - b^{k+1} y_F \\ &= -Q_{FF} s_F - Q_{FA} s_A + e_F - b^{k+1} y_F \end{aligned} \tag{23}$$

With (23), the last two terms of (18) become

$$\begin{aligned} & s_B^T (Q\alpha^k)_B - e_B^T s_B \\ = \; & s_F^T (Q\alpha^k)_F - e_F^T s_F + s_A^T ((Q(\alpha^k + s))_A + b^{k+1} y_A - e_A) - s_A^T (Qs)_A - b^{k+1} y_A^T s_A \\ = \; & s_F^T (Q\alpha^k)_F - e_F^T s_F + s_A^T ((Q(\alpha^k + s))_A + b^{k+1} y_A - e_A) - s_A^T (Q_{AF} s_F + Q_{AA} s_A) - \\ & b^{k+1} y_A^T s_A \\ = \; & -s_F^T Q_{FF} s_F - s_F^T Q_{FA} s_A - b^{k+1} y_B^T s_B + s_A^T ((Q(\alpha^k + s))_A + b^{k+1} y_A - e_A) - \\ & s_A^T (Q_{AF} s_F + Q_{AA} s_A). \end{aligned} \tag{24}$$

If $\alpha_i^k + s_i = 0$, then $s_i \le 0$ and if $\alpha_i^k + s_i = C$, then $s_i \ge 0$. Hence from (21) and (22)

$$s_A^T ((Q(\alpha^k + s))_A + b^{k+1} y_A - e_A) \le 0. \tag{25}$$

With (24), (25),

$$\frac{1}{2} s_B^T Q_{BB} s_B = \frac{1}{2} s_F^T Q_{FF} s_F + s_F^T Q_{FA} s_A + \frac{1}{2} s_A^T Q_{AA} s_A,$$

and $y_B^T s_B = 0$, (18) becomes

$$
\begin{aligned}
& -\frac{1}{2}s_F^T Q_{FF} s_F - \frac{1}{2}s_A^T Q_{AA} s_A - s_F Q_{FA} s_A + s_A^T((Q(\alpha^k + s))_A + b^{k+1} y_A - e_A) \\
\leq \; & -\frac{1}{2}\begin{bmatrix} s_F^T \; s_A^T \end{bmatrix} \begin{bmatrix} Q_{FF} & Q_{FA} \\ Q_{AF} & Q_{AA} \end{bmatrix} \begin{bmatrix} s_F \\ s_A \end{bmatrix} \\
\leq \; & -\frac{\sigma}{2}\|s_B\|^2 = -\frac{\sigma}{2}\|s\|^2.
\end{aligned} \tag{26}
$$

■

From now on we consider any convergent subsequence $\{\alpha^k\}, k \in \mathcal{K}$ and $\lim_{k\to\infty, k\in\mathcal{K}} \alpha^k = \bar{\alpha}$. We then have the following lemma:

Lemma III.3 *For any given positive integer s, the sequence $\{\alpha^{k+s}\}, k \in \mathcal{K}$ converges to $\bar{\alpha}$. In addition, $\{y_i \nabla f(\alpha^{k+s})_i\}$ converges to $y_i \nabla f(\bar{\alpha})_i$, for $i = 1, \ldots, l$.*

Proof: First we know that $\{f(\alpha^k)\}$ is a decreasing sequence. Since $0 \leq \alpha_i \leq C, i = 1, \ldots, l$, the feasible region of (1) is a compact set. Thus we know that $\{f(\alpha^k)\}$ converges to a finite number.

Then for the subsequence $\{\alpha^{k+1}\}, k \in \mathcal{K}$, from Lemma III.2 we have

$$
\begin{aligned}
& \lim_{k\to\infty} \|\alpha^{k+1} - \bar{\alpha}\| \\
\leq \; & \lim_{k\to\infty} (\|\alpha^{k+1} - \alpha^k\| + \|\alpha^k - \bar{\alpha}\|) \\
\leq \; & \lim_{k\to\infty} \left(\sqrt{\frac{2}{\sigma}(f(\alpha^k) - f(\alpha^{k+1}))} + \|\alpha^k - \bar{\alpha}\|\right) \\
= \; & 0.
\end{aligned}
$$

Thus

$$
\lim_{k\to\infty, k\in\mathcal{K}} \alpha^{k+1} = \bar{\alpha}.
$$

From $\{\alpha^{k+1}\}$ we can prove $\lim_{k\to\infty, k\in\mathcal{K}} \alpha^{k+2} = \bar{\alpha}$ too. Therefore, $\lim_{k\to\infty, k\in\mathcal{K}} \alpha^{k+s} = \bar{\alpha}$ for any given s.

The results on $\{y_i \nabla f(\alpha^{k+s})_i\}$ follows from the continuity of $f(\alpha)$. ■

Next we discuss some observations which help to develop techniques for proving the convergence of the decomposition method. If $\hat{\alpha}$ is an optimal solution of (1), it satisfies the following KKT condition: there is a number b such that

$$\begin{aligned} \nabla f(\hat{\alpha})_i + by_i &\geq 0 && \text{if } \hat{\alpha}_i = 0, \\ \nabla f(\hat{\alpha})_i + by_i &\leq 0 && \text{if } \hat{\alpha}_i = C, \\ \nabla f(\hat{\alpha})_i + by_i &= 0 && \text{if } 0 < \hat{\alpha}_i < C. \end{aligned} \tag{27}$$

For any scalar α_i, we can consider two situations

$$0 < \alpha_i < C \text{ or } (\alpha_i = C \text{ and } y_i = 1) \text{ or } (\alpha_i = 0 \text{ and } y_i = -1), \tag{28}$$

$$0 < \alpha_i < C \text{ or } (\alpha_i = C \text{ and } y_i = -1) \text{ or } (\alpha_i = 0 \text{ and } y_i = 1). \tag{29}$$

Then the KKT condition (27) can be rewritten as

$$\begin{aligned} y_i \nabla f(\hat{\alpha})_i + b &\geq 0 && \text{if } \hat{\alpha}_i \text{ satisfies (29)}, \\ y_i \nabla f(\hat{\alpha})_i + b &\leq 0 && \text{if } \hat{\alpha}_i \text{ satisfies (28)}. \end{aligned} \tag{30}$$

Note that (28) ((29)) is the condition in Algorithm I.2 where α_i^k can be a candidate for selection from the top (bottom) of the sorted list of $y_i \nabla f(\alpha^k)_i, i = 1, \ldots, l$. In the following we shall refer a variable α_i as a

"top" candidate: if it satisfies (28),

"top only" candidate: if it satisfies $(\alpha_i = C$ and $y_i = 1)$ or $(\alpha_i = 0$ and $y_i = -1)$,

"bottom" candidate: if it satisfies (29), or

"bottom only" candidate: if it satisfies $(\alpha_i = C$ and $y_i = -1)$ or $(\alpha_i = 0$ and $y_i = 1)$.

From Assumption II.1, $C > 0$ so the following two statements are equivalent:

$$\alpha_i \text{ is a "top only" candidate } \equiv \alpha_i \text{ is not a "bottom" candidate.}$$

Therefore, once α_i^k is a "top only" candidate, next time when it is selected, in Algorithm I.2, it must be picked from the top of the sorted list.

It can be clearly seen that the KKT condition (30) implies that all "top" candidates have the same or smaller $y_i \nabla f(\hat{\alpha})_i$ than "bottom" candidates. Therefore, when applying Algorithm I.2 to problem (3) of an optimal solution, except those elements with $y_i \nabla f(\hat{\alpha}) + b = 0$, we cannot do any selection.

For free variables, their $y_i \nabla f(\hat{\alpha})_i$ are equal. On the other hand, if $\hat{\alpha}_i$ are at bounds, their $y_i \nabla f(\hat{\alpha})_i$ are usually different. This leads us to suspect that in final iterations, all bounded α_i's associated $y_i \nabla f(\alpha)_i$ are already in correct places of the sorted list of $y_i \nabla f(\hat{\alpha})_i, i = 1, \ldots, l$.

Of course is possible that $\nabla f(\hat{\alpha})_i + b y_i = 0$ even if $\hat{\alpha}_i$ is at a bound. This is the so called "degenerate" case in optimization terminology. For degenerate or free variables, $y_i \nabla f(\hat{\alpha})_i$ are all equal. We will focus on analyzing this group of variables. Indeed we will show that in final iterations, only indices from this particular group are still under consideration.

Next we show a lemma related to the KKT condition of sub-problems:

Lemma III.4 *For an optimal solution α_B of (2), there is a number b such that if $\alpha_i, i \in B$, satisfies (28), then*

$$y_i \nabla f(\alpha)_i + b \leq 0,$$

and if $\alpha_i, i \in B$, satisfies (29), then

$$y_i \nabla f(\alpha)_i + b \geq 0.$$

We then need two technical lemmas:

Lemma III.5 *If $\bar{\alpha}_i$ satisfies (28) ((29)), then for any given positive integer s, after $k \in \mathcal{K}$ is large enough, $\alpha_i^k, \alpha_i^{k+1}, \cdots, \alpha_i^{k+s}$ all satisfy (28) ((29)). In other words, if $\bar{\alpha}_i$ is a "top" ("bottom") candidate, then after $k \in \mathcal{K}$ is large enough, $\alpha_i^k, \alpha_i^{k+1}, \cdots, \alpha_i^{k+s}$ are all "top" ("bottom") candidates. In addition, if*

$$y_{i_1} \nabla f(\bar{\alpha})_{i_1} > y_{i_2} \nabla f(\bar{\alpha})_{i_2}, \tag{31}$$

then after $k \in \mathcal{K}$ is large enough, for any $\bar{k} \in \{k, k+1, \ldots, k+s-1\}$, if i_1 and i_2 are both in the working set of the $\bar{k}$th iteration, it is impossible to have $\alpha_{i_1}^{\bar{k}+1}$ and $\alpha_{i_2}^{\bar{k}+1}$ satisfy (28) and (29), respectively.

Proof: The first result immediately follows from Assumption II.1, Lemma III.3, and the definition of (28) and (29).

For the second result of this lemma, we assume that it is possible that both $\alpha_{i_1}^{\bar{k}+1}$ and $\alpha_{i_2}^{\bar{k}+1}$ satisfy (28) and (29), respectively. Since $\alpha_B^{\bar{k}+1}$ is an optimal solution of (2), from Lemmas III.4, if $\alpha_{i_1}^{\bar{k}+1}$ satisfies (28), there is a $b^{\bar{k}+1}$ such that

$$y_{i_1}\nabla f(\alpha^{\bar{k}+1})_{i_1} + b^{\bar{k}+1} \le 0. \tag{32}$$

On the other hand, if $\alpha_{i_2}^{\bar{k}+1}$ satisfies (29), then

$$y_{i_2}\nabla f(\alpha^{\bar{k}+1})_{i_2} + b^{\bar{k}+1} \ge 0. \tag{33}$$

Thus (32) and (33) imply

$$y_{i_1}\nabla f(\alpha^{\bar{k}+1})_{i_1} \le y_{i_2}\nabla f(\alpha^{\bar{k}+1})_{i_2}$$

which contradicts to (31) when $\bar{k}$ is large enough. ■

Lemma III.6 *Consider the following problem:*

$$\begin{aligned} \min \quad & \nabla f(\bar{\alpha})^T d \\ & -1 \le d_i \le 1, y^T d = 0, \\ & d_i \ge 0, \ \textit{if } \bar{\alpha}_i = 0, \qquad d_i \le 0, \ \textit{if } \bar{\alpha}_i = C, \\ & |\{d_i \mid d_i \ne 0\}| \le q. \end{aligned}$$

Assume Algorithm I.2 is used to solve this problem. If i_1 (i_2) is the first element selected from the top (bottom) of the sorted list of $y_i\nabla f(\bar{\alpha})_i, i = 1, \ldots, l$, then

$$y_{i_1}\nabla f(\bar{\alpha})_{i_1} = y_{i_2}\nabla f(\bar{\alpha})_{i_2}.$$

Proof: If the result is wrong, then

$$y_{i_1}\nabla f(\bar{\alpha})_{i_1} > y_{i_2}\nabla f(\bar{\alpha})_{i_2}. \tag{34}$$

Define

$$I_1 \equiv \{i \mid y_i\nabla f(\bar{\alpha})_i \ge y_{i_1}\nabla f(\bar{\alpha})_{i_1}\}$$

and

$$I_2 \equiv \{i \mid y_i\nabla f(\bar{\alpha})_i \le y_{i_2}\nabla f(\bar{\alpha})_{i_2}\}.$$

From (34), $I_1 \cap I_2 = \emptyset$.

Since i_1 (i_2) is the first element selected from the top (bottom) of the sorted list, $\bar{\alpha}_{i_1}$ ($\bar{\alpha}_{i_2}$) satisfies (28) ((29)). After $k \in \mathcal{K}$ is large enough, from Lemma III.5, $\alpha_{i_1}^k, \alpha_{i_1}^{k+1}, \cdots, \alpha_{i_1}^{k+2l}$ are all "top" candidates, where l is the length of each vector α (i.e. the number of variables of (1)). In addition, $\alpha_{i_2}^k, \alpha_{i_2}^{k+1}, \cdots, \alpha_{i_2}^{k+2l}$ are all "bottom" candidates. Then in each $\bar{k}$ of kth, $(k+1)$st, ... , $(k+2l-1)$st iterations, i_1 and i_2 can not both be selected because of (34) and Lemma III.5.

We then claim that if i_1 is not selected at the $\bar{k}$th iteration, then all "top" candidates selected are from I_1. Since $\bar{k}$ is large enough, for any $\alpha_i^{\bar{k}}$, $i \notin I_1$, which is a "top" candidate,

$$y_i \nabla f(\bar{\alpha})_i < y_{i_1} \nabla f(\bar{\alpha})_{i_1}$$

implies that i can not be chosen earlier then i_1. Similarly, if i_2 is not selected at the $\bar{k}$th iteration, then all "bottom" candidates selected are from I_2.

Now for the $\bar{k}$th iteration, we consider three situations:

Case 1: Neither i_1 nor i_2 is selected: Then all "top" ("bottom") candidates selected are in $I_1(I_2)$. For any $i \in I_1$ and $j \in I_2$ selected in the $\bar{k}$th iteration, from Lemma III.5, at the next iteration, either $\alpha_{i_1}^{\bar{k}+1}$ becomes a "bottom only" element or $\alpha_{i_2}^{\bar{k}+1}$ becomes a "top only" element. Therefore, there are two cases

Case 1-1: All elements selected from I_1 become "bottom only:" Then the number of "bottom only" variables in I_1 is increased by at least one. On the other hand, since $I_1 \cap I_2 = \emptyset$, all variables selected from I_2 are "bottom" elements. Hence the number of "top only" variables in I_2 is at least the same.

Case 1-2: All elements selected from I_2 become "top only:" Similarly, the number of "top only" variables in I_2 is increased by at least one, while the number of "bottom only" variables in I_1 is at least the same.

Case 2: Only i_1 is selected: As i_2 is not selected, all "bottom" elements selected are in I_2. Since i_1 is selected and $\alpha_{i_1}^{\bar{k}+1}$ is a "top" candidate, all "bottom" elements become "top only." Therefore, the number of "top only" variables in I_2 increases by at least one. On the other hand, the number of "bottom only" variables in I_1 is at least the same.

Case 3: Only i_2 is selected: Similar to case 2, the number of "bottom only" variables in I_1 increases at least one and the number of "top only" variables in I_2 is at least the same.

Therefore, in at most l iterations, either all elements in I_1 become "bottom only" or all elements in I_2 become "top only." If I_1 reaches "bottom only" first, from Assumption II.1, for later iterations, elements in I_1 are not "top" candidates so i_1 must be selected. Therefore, we only have case 2 left. Then after at most another l iterations, all I_2 are "top only." Therefore, we must have a $\bar{k} \in \{k, k+1, \ldots, k+2l\}$ such that both i_1 and i_2 are selected. This contradicts to Lemma III.5. ■

Finally, the main theorem is as follows:

Theorem III.7 *Any limit point of $\{\alpha^k\}$ is a global minimum of (1).*

Proof: Assume $\bar{\alpha}$ is the limit point of any convergent subsequence $\{\alpha_k\}, k \in \mathcal{K}$. If $\bar{\alpha}$ is not an optimal solution of (1), from Theorem II.4, the following problem has a nonzero solution:

$$\begin{aligned} \min \quad & \nabla f(\bar{\alpha})^T d \\ & -1 \leq d_i \leq 1, y^T d = 0, \\ & d_i \geq 0, \text{ if } \bar{\alpha}_i = 0, \qquad d_i \leq 0, \text{ if } \bar{\alpha}_i = C, \\ & |\{d_i \mid d_i \neq 0\}| \leq q. \end{aligned} \tag{35}$$

Then from Lemma III.6, only elements with the same $y_i \nabla f(\bar{\alpha})_i$ can have nonzero d_i. Assume B contains such indices. Then

$$\nabla f(\bar{\alpha})^T d = \sum_{i \in B} y_i^2 \nabla f(\bar{\alpha})_i d_i = (y_i \nabla f(\bar{\alpha})_i) y_B^T d_B = 0.$$

This contradicts to the assumption that (35) has a nonzero solution. ■

IV. Extensions

Consider a general problem with the following form:

$$\begin{aligned} \min \quad & \frac{1}{2}\alpha^T Q \alpha + p^T \alpha \\ & y^T \alpha = \Delta, \\ & l_i \leq \alpha_i \leq u_i, i = 1, \ldots, l, \end{aligned} \tag{36}$$

where $-\infty < l_i < u_i < \infty, i = 1, \ldots, l$, Q is any symmetric positive semi-definite matrix satisfying Assumption III.1, and $y_i = \pm 1, i = 1, \ldots, l$. The convergence proof described in the previous section is still valid if Algorithm I.1 with the following generalized working set selection is used for solving (36):

$$
\begin{aligned}
\min \quad & \nabla f(\alpha^k)^T d \\
& y^T d = 0, \ -1 \leq d_i \leq 1, i = 1, \ldots, l, \\
& d_i \geq 0, \text{ if } (\alpha^k)_i = l_i, \qquad d_i \leq 0, \text{ if } (\alpha^k)_i = u_i, \\
& |\{d_i \mid d_i \neq 0\}| \leq q.
\end{aligned} \tag{37}
$$

It can be seen that $y_i = \pm 1$ plays an important role here. Algorithm I.2 is not valid for solving (37) if this condition does not hold. In addition, in the convergence proof we specifically utilize many properties of Algorithm I.2 (e.g. we consider the sorted list of $y_i \nabla f(\bar{\alpha})_i$) so the condition $y_i = \pm 1$ is also used. In [10], the authors handled a more generalized problem where the only restriction on y_i is $y_i \neq 0$.

Problem (36) covers most SVM formulations. For example, given a set of data points $\{(x_1, z_1), \ldots, (x_l, z_l)\}$ such that $x_i \in R^n$ is an input and $z_i \in R^1$ is a target output, the usual form of support vector regression is as follows:

$$
\begin{aligned}
& \min \frac{1}{2}(\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{l} z_i (\alpha_i - \alpha_i^*) \\
& \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0, 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, \ldots, l,
\end{aligned} \tag{38}
$$

where $Q_{ij} = \phi(x_i)^T \phi(x_j)$.

We can rewrite (38) as

$$
\begin{aligned}
& \min \frac{1}{2} \begin{bmatrix} \alpha^T, (\alpha^*)^T \end{bmatrix} \begin{bmatrix} Q & -Q \\ -Q & Q \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha^* \end{bmatrix} + \begin{bmatrix} \epsilon e^T - z^T, \epsilon e^T + z^T \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha^* \end{bmatrix} \\
& y^T \begin{bmatrix} \alpha \\ \alpha^* \end{bmatrix} = 0, 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, \ldots, l,
\end{aligned} \tag{39}
$$

where y is a $2l$ by 1 vector with $y_i = 1, i = 1, \ldots, l$ and $y_i = -1, i = l+1, \ldots, 2l$. (39) is in the form of (36) so Algorithms I.1 and I.2 can be applied.

However, using Algorithms I.1 and I.2 for (39) is a little different from existing decomposition methods for regression. Note that though (38) is a problem with $2l$ variables, it has very special structures. For example, the KKT condition implies that at an optimal solution of (38), $\alpha_i\alpha_i^* = 0$. Early work on SVM regression (e.g. [24], [12], [13], [6]) all tried to take advantage of these structures and focused on problem (38). Except [13] they mainly consider selecting two elements as the working set in each iteration. Some characteristics of their methods are:

1. In each iteration, two indices i_1 and i_2 are selected from $\{1, \ldots, l\}$.
2. To keep $\alpha_i^k(\alpha^*)_i^k = 0$, they solve a sub-problem with four variables $\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_1}^*$, and $\alpha_{i_2}^*$. That is, it is like that they use $q = 4$ in Algorithm I.1 with a different working set selection from Algorithm I.2.

If Algorithms I.1 and I.2 with $q = 2$ are directly used for (39), two indices are selected from $\{1, \ldots, 2l\}$ and a sub-problem (2) with two variables is solved. The advantage of working on (39) is that a generalized implementation can be directly used for both classification and regression. However, a possible shortcoming is that special structures of (38) are not considered so there may have computational overheads. Surprisingly we will show that $\alpha_i^k(\alpha^*)_i^k = 0$ still holds if Algorithms I.1 and I.2 are directly applied to solve (39). Another issue is on the Hessian $\bar{Q} = \left[\begin{smallmatrix} Q & -Q \\ -Q & Q \end{smallmatrix}\right]$ of the objective function of (39). Now $\bar{Q}$ is only positive semidefinite so it is unlikely that Assumption III.1 can be true. In the following theorem we will show that if only Q instead of $\bar{Q}$ satisfies Assumption III.1, the convergence for solving (39) follows.

Theorem IV.1 *If Algorithms I.1 and I.2 are used for solving (39) and the initial solution is zero, then $\alpha_i^k(\alpha^*)_i^k = 0, i = 1, \ldots, l$ for all k. In addition, if Q satisfies Assumption III.1, $\left[\begin{smallmatrix} \alpha^k \\ (\alpha^*)^k \end{smallmatrix}\right]$ converges to an optimal solution of (38).*

Proof:

We prove the first result by the mathematical induction. It is true that if the initial solution is zero, for the first iteration, $\alpha_i^1(\alpha^*)_i^i = 0, i = 1, \ldots, l$. Assume the result is true for the kth iteration and we will prove $\alpha_i^{k+1}(\alpha^*)_i^{k+1} = 0, i = 1, \ldots, l$.

We consider three situations in the kth iteration:

1. In Algorithm I.2, both i and $i + l$ are selected in the working set: Then from the KKT condition of the sub-problem (2), $\alpha_i^{k+1}(\alpha^*)_i^{k+1} = 0$.

2. Only i but not $i+l$ is selected in the working set and $\alpha_i^k = 0$: For this case, $d_i = 1$ after solving (3). Since $y_i d_i = 1$, we realize that in Algorithm I.2, the index i is selected from the bottom of the sorted list of $y_i \nabla f(\alpha^k, (\alpha^*)^k)_i, i = 1, \ldots, 2l$. However, we also have

$$\begin{aligned} y_i \nabla f(\alpha^k, (\alpha^*)^k)_i &= (Q(\alpha^k - (\alpha^*)^k))_i + \epsilon - z_i \\ \geq \;\; & (Q(\alpha^k - (\alpha^*)^k))_i - \epsilon - z_i = y_{i+l} \nabla f(\alpha^k, (\alpha^*)^k)_{i+l}. \end{aligned}$$

In other words, index $i+l$ is closer than i to the bottom of the sorted list. Therefore, if $i+l$ is not selected, index $i+l$ is not a "bottom" candidate so $(\alpha^*)_i^k$ does not satisfy (29). As $y_{i+l} = -1$, $(\alpha^*)_i^k = 0$. Since $i+l$ is not selected in the kth iteration, $(\alpha^*)_i^{k+1} = (\alpha^*)_i^k = 0$ so $\alpha_i^{k+1}(\alpha^*)_i^{k+1} = 0$.

3. Only i but not $i+l$ is selected in the working set and $\alpha_i^k > 0$: Then $\alpha_i^k(\alpha^*)_i^k = 0$ implies $(\alpha^*)_i^k = 0$. Therefore, $(\alpha^*)_i^{k+1} = 0$ and $\alpha_i^{k+1}(\alpha^*)_i^{k+1} = 0$.

When only $i+l$ but not i is selected, the situation is similar. Thus we have finished the proof that $\alpha_i^k(\alpha^*)_i^k = 0, i = 1, \ldots, l$, for all k.

Next we switch to the second goal of this theorem. Now the Hessian of the objective function of (39) is $\bar{Q} = \left[\begin{smallmatrix} Q & -Q \\ -Q & Q \end{smallmatrix}\right]$. We remember that Assumption III.1 is needed near Eq. (26) in the proof of Lemma III.2. If we can prove

$$-\frac{1}{2} s_B^T \bar{Q}_{BB} s_B \leq -\frac{\sigma}{2} \|s_B\|^2, \tag{40}$$

where σ is related only to Q, then a condition on Q instead of $\bar{Q}$ is sufficient for the convergence. Thus the main task is to prove that (40) is true.

We define the following disjoint index sets:

$$\begin{aligned} &B_1 = \{i \mid 1 \leq i \leq l, i \in B \text{ and } i+l \in B\}, \qquad B_1^* = \{i+l \mid i \in B_1\}, \\ &B_2 = \{i \mid 1 \leq i \leq l, i \in B \text{ and } i+l \notin B\}, \\ &B_3^* = \{i+l \mid 1 \leq i \leq l, i \notin B \text{ and } i+l \in B\}, \qquad B_3 = \{i \mid i+l \in B_3^*\}. \end{aligned}$$

Then

$$B = B_1 \cup B_1^* \cup B_2 \cup B_3^*.$$

Thus

$$
\begin{aligned}
& s_B^T \bar{Q}_{BB} s_B \\
= \;& [s_{B_1} - s_{B_1^*}, s_{B_2}, s_{B_3^*}] \begin{bmatrix} Q_{B_1B_1} & Q_{B_1B_2} & -Q_{B_1B_3} \\ Q_{B_2B_1} & Q_{B_2B_2} & -Q_{B_2B_3} \\ -Q_{B_3B_1} & -Q_{B_3B_2} & Q_{B_3B_3} \end{bmatrix} \begin{bmatrix} s_{B_1} - s_{B_1^*} \\ s_{B_2} \\ s_{B_3^*} \end{bmatrix}. \qquad (41)
\end{aligned}
$$

Since $\begin{bmatrix} Q_{B_1B_1} & Q_{B_1B_2} & -Q_{B_1B_3} \\ Q_{B_2B_1} & Q_{B_2B_2} & -Q_{B_2B_3} \\ -Q_{B_3B_1} & -Q_{B_3B_2} & Q_{B_3B_3} \end{bmatrix}$ has the same eigenvalues as $\begin{bmatrix} Q_{B_1B_1} & Q_{B_1B_2} & Q_{B_1B_3} \\ Q_{B_2B_1} & Q_{B_2B_2} & Q_{B_2B_3} \\ Q_{B_3B_1} & Q_{B_3B_2} & Q_{B_3B_3} \end{bmatrix}$, which is a square sub-matrix of Q, and $|B_1 \cup B_2 \cup B_3| \leq q$, we have

$$
\begin{aligned}
& -\frac{1}{2} s_B^T \bar{Q}_{BB} s_B \\
\leq \;& -\frac{\sigma}{2} \| \begin{bmatrix} s_{B_1} - s_{B_1^*} \\ s_{B_2} \\ s_{B_3^*} \end{bmatrix} \|^2 \leq -\frac{\sigma}{2} \| \begin{bmatrix} s_{B_1} \\ s_{B_1^*} \\ s_{B_2} \\ s_{B_3^*} \end{bmatrix} \|^2 = -\frac{\sigma}{2} \|s_B\|^2,
\end{aligned}
$$

where $\sigma = \min_I(\min(\mathrm{eig}(Q_{II})))$, and I is any subset of $\{1, \ldots, l\}$ with $|I| \leq q$. Note that $\|s_{B_1} - s_{B_1^*}\|^2 \geq \| \begin{bmatrix} s_{B_1} \\ s_{B_1^*} \end{bmatrix} \|^2$ is because of the following reasons: Since $\alpha_i^{k+1} (\alpha^*)_i^{k+1} = 0$, we consider two situations:

1. $\alpha_i^{k+1} = 0$: Then if $\alpha_i^k = 0$, $s_i = 0$ so $s_i s_{i+l} \leq 0$. On the other hand, if $\alpha_i^k > 0$, $(\alpha^*)_i^k = 0$. Hence $s_i \leq 0$ and $s_{i+l} \geq 0$ imply $s_i s_{i+l} \leq 0$.

2. $(\alpha^*)_i^{k+1} = 0$: Similarly, $s_i s_{i+l} \leq 0$.

Therefore, we have $-s_{B_1}^T s_{B_1^*} \geq 0$ so

$$
\|s_{B_1} - s_{B_1^*}\|^2 = \| \begin{bmatrix} s_{B_1} \\ s_{B_1^*} \end{bmatrix} \|^2 - 2 s_{B_1}^T s_{B_1^*} \geq \| \begin{bmatrix} s_{B_1} \\ s_{B_1^*} \end{bmatrix} \|^2.
$$

■

Recent implementation using Algorithms I.1 and I.2 with $q = 2$ for (39) are LIBSVM (version 2.0) [3] and SVMTorch [5].

Note that it is also possible to extend convergence results in this section for algorithms used in [12], [13], [24] but here we will not get into details.

We then briefly discuss two other SVM formulations: one-class SVM and ν-SVM. For one-class SVM [22], the formulation is already in the form of (36). For ν-SVM [23], it has two linear constraints so is not covered by the algorithm and proof here. However, a variant by removing one linear constraint also generates a formulation of (36). In [4], the algorithm of SVM^{light} was adopted to solve this modified ν-SVM problem.

V. Conclusions and Discussions

In this section we give some notes about the convergence proof. The property that (2) is exactly solved is used both in Lemmas III.2 and III.4. This confirms the conjectures in Section I where we think that an optimal solution of (2) makes a difference from the original Zoutendijk's method.

It is unfortunate that we need Assumption III.1 for the proof. We hope that this gap can be filled sometime in the future.

The convergence proof also suggests a possible way to improve the implementation. In final iterations, as the order of sorting $y_i \nabla f(\alpha^*)_i, i = 1, \ldots, l$ is about fixed, it might be possible to consider fewer elements on the working set selection.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grant NSC 89-2213-E-002-013. The author thanks Sathiya Keerthi for many help comments.

References

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming : theory and algorithms*. Wiley, second edition, 1993.

[2] C.-C. Chang, C.-W. Hsu, and C.-J. Lin. The analysis of decomposition methods for support vector machines. *IEEE Trans. Neural Networks*, 11(4):1003–1008, 2000.

[3] C.-C. Chang and C.-J. Lin. Libsvm 2.0: Solving different support vector formulations, 2000. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] C.-C. Chang and C.-J. Lin. Training ν-support vector classifiers: Theory and algorithms. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2000. Submitted to Neural Computation.

[5] R. Collobert and S. Bengio. SVMTorch: A support vector machine for large-scale regression and classification problems, 2000. Available at `http://www.idiap.ch/learning/SVMTorch.html`.

[6] G. W. Flake and S. Lawrence. Efficient SVM regression training with SMO. Technical report, NEC Research Institute, 1999.

[7] T.-T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proceeding of 15th Intl. Conf. Machine Learning*. Morgan Kaufman Publishers, 1998.

[8] C.-W. Hsu and C.-J. Lin. A simple decomposition method for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 1999.

[9] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

[10] S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. Technical Report CD-00-01, Department of Mechanical and Production Engineering, National University of Singapore, Singapore, 2000. Submitted to Machine Learning.

[11] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. Technical report, Department of Mechanical and Production Engineering, National University of Singapore, 1999. To appear in Neural Computation.

[12] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. Improvements to SMO algorithm for SVM regression. Technical Report CD-99-16, Department of Mechanical and Production Engineering, National University of Singapore, 1999.

[13] P. Laskov. An improved decomposition algorithm for regression support vector machines. In *Workshop on Support Vector Machines, NIPS99*, 1999.

[14] O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Trans. Neural Networks*, 10(5):1032–1037, 1999.

[15] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[16] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of CVPR'97*, 1997.

[17] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

[18] M. J. D. Powell. On search directions for minimization. *Math. Programming*, 4:193–201, 1973.

[19] S. Rüping. mySVM - another one of those support vector machines, 2000. Software available at `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`.

[20] C. Saunders, M. O. Stitson, J. Weston, L. Bottou, B. Schölkopf, and A. Smola. Support vector machine reference manual. Technical Report CSD-TR-98-03, Royal Holloway, University of London, Egham, UK, 1998.

[21] B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, 1998.

[22] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report 99-87, Microsoft Research, 1999.

[23] B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207 – 1245, 2000.

[24] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Neuro COLT Technical Report TR-1998-030, Royal Holloway College, 1998.

[25] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 1995.

[26] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, NY, 1998.

[27] P. Wolfe. On the convergence of gradient methods under constraint. *IBM J. Research and Development*, 16:407–411, 1972.

[28] G. Zoutendijk. *Methods of feasible directions.* Elsevier, 1960.