Feasible Direction Decomposition Algorithms for Training Support Vector Machines

Pavel Laskov

Department of Computer and Information Sciences
102 Smith Hall
University of Delaware
Newark, DE 19718
laskov@cis.udel.edu

Abstract. The article presents a general view of a class of decomposition algorithms for training Support Vector Machines (SVM) motivated by the method of feasible directions. Such an algorithm for the pattern recognition SVM has been proposed by Joachims [3]. Its extension to the regression SVM – the maximal inconsistency algorithm – has been recently presented by the author [5]. A detailed account of both algorithms is carried out, complemented by theoretical investigation of the relatioship between the two algorithms. It is proved that the two algorithms are equivalent for the pattern recognition SVM, and the feasible direction interpretation of the maximal inconsistency algorithm is given for the regression SVM. The experimental results demonstrate an order of magnitude decrease of training time in comparision with training without decomposition, and provide experimental evidence of linear convergence rate of the decomposition algorithms.

1 Introduction

Computational complexity of SVM training algorithms is attracting increasing interest, as applications of SVM extend to problems of larger and larger size. Requests have been recently made for algorithms capable of handling problems containing 10^5 – 10^6 examples [14]. The basic training algorithm [16] involves a solution to a quadratic programming problem the size of which depends on the size of the training data set. For the training data set of size l, the pattern recognition SVM quadratic program has l variables with non-negativity constraints, l inequality constraints and one equality constraint. The regression SVM quadratic program has $2l$ variables with non-negativity constraints, $2l$ inequality constraints and one equality constraint. In addition, the basic formulation of the SVM training algorithm requires storage of the kernel matrix of size $l \times l$ or $2l \times 2l$ respectively. Thus, the running time and storage of the kernel matrix are the two main optimization-related bottlenecks of SVM training algorithms[1].

[1] Another possible bottleneck is computation of the kernel matrix. If real dimension of data points is large, this time may become comparable with training time. However

First SVM implementations used general purpose optimization packages (such as MINOS, LOQO and others) that were not designed for problems of such size. It was soon discovered that these packages were not suitable for solutions to problems involving more than a few hundred examples. The early special-purpose methods, proposed to speed-up training, have not brought much relief. Chunking [15] prescribes iteration through the training data, accumulating support vectors and adding a "chunk" of new data until no more changes to the solution occur. The main problem with this method is that when the percentage of support vectors is high it essentially solves the problem of almost the full size more than once. Another method proposed by Kaufmann [4] modifies traditional optimization algorithms (a combination of Newton's and conjugate gradient methods) to yield the overall complexity of $O(s^3)$ per iteration, where s is the (a priori unknown) number of support vectors. This can be a significant improvement over $O(l^3)$, however the number of support vectors in not guaranteed to be small.

Decomposition was the first practical method for solving large-scale SVM training problems. It was originally proposed for the pattern recognition SVM [9] and subsequently extended to the regression SVM in [8]. The key idea of decomposition is to freeze all but a small number of optimization variables, and to solve a sequence of constant-size problems. The set of variables optimized at a current iterations is denoted as the *working set*. Because the working set is re-optimized, the value of the objective function is improved at each iteration, provided the working set is not optimal before re-optimization. Iteration is stopped when termination criteria, derived from Karush-Kuhn-Tucker (KKT) conditions, are satisfied to a required precision.

Selection of the working set is the most important issue in the decomposition algorithms. First, the provision that the working set must be sub-optimal before re-optimizaiton, is crucial to prevent the algorithm from cycling. Therefore, it is important to have criteria to test sub-optimality of any given working set. Second, working set selection affects the rate of convergence of the algorithm: if sub-optimal working sets are selected more or less at random the algorithm converges very slowly. Finally, working set selection is imporant in theoretical analysis of decomposition algorithms; in particular, in the recent convergence proof by Chang *et. al.* [2].

The original decomposition algorithm essentially addressed only the first issue—the design of its termination criteria in all but pathological cases prevents composition of already optimal working sets. Its implementation featured some unpublished heuristics which provided reasonable convergence speed. Obviously, a formal framework for working set selection was highly desirable. One such framework is the method of feasible directions, proposed in the optimization theory in 1960 by Zoutendijk [18]. The connection between this method and the working set selection problem was first discovered by Joachims in a paper

---

it will be assume in the rest of this article that kernel computation is not the main factor contributing to the complexity of training.

that has drawn wide attention [3]. However, Joachims' algorithm is limited to the pattern recognition case because it uses the fact that the labels are ± 1.

The main goal of the current article is to provide a unified treatment of the working set selection problem within the framework of the method of feasible directions. Specifically, the first two issues raised above are addressed, in a common way for the pattern recognition and the regression SVM:

- a criterion is proposed to identify sub-optimal working sets
- a heuristic, which shares the motivation with the optimality criterion, is proposed for (approximately) optimal working set selection.

The new algorithm, termed the "maximal inconsistency algorithm", is applicable for both pattern recognition and the regression SVM. To further reveal the machinery of the new algorithm, it is shown that for the pattern recognition SVM the new algorithm is equivalent to Joachims' algorithm, and for the regression SVM a similar algorithm exists based on the feasible direction principle. Similar to the pattern recognition case, the maximal inconsistency algorithm satisfies theoretical requirements crucial for the proof of convergence. These relationships allow to classify both algorithms as "feasible direction decomposition" algorithms.

The article is organized as follows. Section 2 provides basic formulations of decomposition algorithms for the pattern recognition and the regression SVM. Section 3 presents the method of feasible directions. Sections 4 and 5 cover the issues related to feasible direction decomposition algorithms for the pattern recognition and the regression SVM respectively. Exposition in these sections is carried out in different fashion. Section 4 is presented in a more or less chronological order. It starts with Joachims' algorithm, fully motivated by the feasible direction method. Then the criterion for testing optimality of a working set is presented, and the maximal inconsistency working set selection rule is derived from this criterion. Finally, equivalence between the maximal inconsistency algorithm and Joachims' algorithm is proven. The order of presentation is reversed for the regression SVM in Section 5. The maximal inconsistency algorithm is introduced first, followed by its interpretation as a feasible direction decomposition algorithm. Experimental results for the regression SVM are presented in Section 6.

2 SVM Decomposition

The classical problem of training the SVM, given the data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ and hence, the kernel matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, can be expressed as the following quadratic program:

$$\begin{array}{rlcl} \text{Maximize} & W(\tilde{\boldsymbol{\alpha}}) & = & \tilde{\mathbf{y}}^T \tilde{\boldsymbol{\alpha}} - \frac{1}{2}\tilde{\boldsymbol{\alpha}}^T D \tilde{\boldsymbol{\alpha}} \\ \text{subject to:} & \mathbf{c}^T \tilde{\boldsymbol{\alpha}} & = & 0 \\ & \tilde{\boldsymbol{\alpha}} - C\mathbf{1} & \leq & \mathbf{0} \\ & \tilde{\boldsymbol{\alpha}} & \geq & \mathbf{0} \end{array} \tag{1}$$

where interpretation of symbolilc components of (1) varies between the pattern recognition and the regression cases. For the pattern recognition SVM,

$$\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}, \quad \tilde{\mathbf{y}} = \mathbf{1}, \quad D = K, \quad \mathbf{c} = \mathbf{y} \tag{2}$$

For the regression SVM,

$$\tilde{\boldsymbol{\alpha}} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} - \epsilon\mathbf{1} \\ -\mathbf{y} - \epsilon\mathbf{1} \end{bmatrix}, \quad D = \begin{bmatrix} K & -K \\ -K & K \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{1} \\ -\mathbf{1} \end{bmatrix} \tag{3}$$

Details of formulation of SVM training problems can be found in [16], [17], [11], [13].

As it was mentioned in the introduction, the main idea of decomposition is to allow only a subset of optimization variables to change weights at a current iteration. The iteration process is repeated until the termination conditions are met.

Let $\tilde{\boldsymbol{\alpha}}_B$ denote the variable included in current working set (of fixed size q) and let $\tilde{\boldsymbol{\alpha}}_N$ denote the rest of the variables. The corresponding parts of vectors $\mathbf{c}$ and $\tilde{\mathbf{y}}$ will also bear subscripts N and B. Matrix D will be partitioned into D_{BB}, $D_{BN} = D_{NB}^T$ and D_{NN}. It will be further required that, for the regression SVM, both α_i and α_i^* are either included in or omitted from the working set.[2]

Optimization of the working set turns out to be also a quadratic program. This can be seen by rearranging the terms of the objective function and the equality constraint in (1), and dropping the terms independent of $\tilde{\boldsymbol{\alpha}}_B$ from the objective. The resulting quadratic program (sub-problem) is formulated as follows:

$$\begin{array}{lll} \text{Maximize} & W_B(\tilde{\boldsymbol{\alpha}}_B) & = (\tilde{\mathbf{y}_B}^T - \tilde{\boldsymbol{\alpha}}_N^T D_{NB})\tilde{\boldsymbol{\alpha}}_B - \frac{1}{2}\tilde{\boldsymbol{\alpha}}_B^T D_{BB}\tilde{\boldsymbol{\alpha}}_B \\ \text{subject to:} & \mathbf{c}_B^T\tilde{\boldsymbol{\alpha}}_B + \mathbf{c}_N^T\tilde{\boldsymbol{\alpha}}_N & = 0 \\ & \tilde{\boldsymbol{\alpha}}_B - C\mathbf{1} & \leq \mathbf{0} \\ & \tilde{\boldsymbol{\alpha}}_B & \geq \mathbf{0} \end{array} \tag{4}$$

The termination conditions of the decomposition algorithm are derived from the KKT conditions ([9], [8]). It is best to consider them separately for the two types of SVM.

For the pattern recognition SVM, let

$$g_i = \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + b \tag{5}$$

where b is the threshold of SVM computed as [3]:

$$b = \frac{1}{|\mathrm{SV}|} \sum_{i \in \mathrm{SV}} \left[y_i - \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right] \tag{6}$$

[2] Strictly speaking this rule is not required in decomposition, however it facilitates formulation of sub-problems to be solved at each iteration.

Then the point satisfies the KKT provided:

$$\begin{aligned} y_i g_i = 1, &\quad \text{if } 0 < \alpha_i < C \\ y_i g_i \leq 1, &\quad \text{if } \alpha_i = C \\ y_i g_i \geq 1, &\quad \text{if } \alpha_i = 0 \end{aligned} \tag{7}$$

For the regression SVM, let

$$g_i = \sum_{j=1}^{l} (\alpha_j - \alpha_j^*) K(\mathbf{x}_i, \mathbf{x}_j) + b \tag{8}$$

where

$$b = \frac{1}{|\mathrm{SV}|} \sum_{i \in \mathrm{SV}} \left[y_i - \sum_{j=1}^{l} (\alpha_j - \alpha_j^*) K(\mathbf{x}_i, \mathbf{x}_j) - \epsilon \right] \tag{9}$$

Then the point satisfies the KKT conditions provided:

$$\begin{aligned} |g_i - y_i| = \epsilon, &\quad \text{if } 0 < \alpha_i < C \text{ and } 0 < \alpha_i^* < C \\ |g_i - y_i| \geq \epsilon, &\quad \text{if } \alpha_i = C \text{ or } \alpha_i^* = C \\ |g_i - y_i| \leq \epsilon, &\quad \text{if } \alpha_i = 0 \text{ and } \alpha_i^* = 0 \end{aligned} \tag{10}$$

Conditions (7) or (10) are checked for *individual points*: if, at a given iteration, one of them is violated for some point, this point is exchanged with some point in the current working set. Thus, the new working set is sub-optimal, and a strict improvement of the overall objective function is achieved.

The problem with using conditions (7) or (10) is that they require knowledge of the threshold b which is also the Lagrange multiplier for the equality constraint of the SVM problem. The formulas given in (6) and (9) require that the set SV of *unbounded* support vectors is non-empty. Usually this assumption is true, however it cannot be guaranteed. While a simple trick can rectify the problem for the pattern recognition SVM, by letting

$$b = -\frac{m + M}{2} \tag{11}$$

where $m = \min_{i \in SV} \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)$ and $M = \max_{i \in SV} \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)$, a similar trick doesn't work for the regression SVM.

Overcoming this problem has been the initial motivation for the new termination condition proposed in this article. Instead of individual points, optimality of the entire working set will be considered. The new condition is described in detail in Section 4.2.

Intelligent selection of working sets is possible by utilizing ideas of the method of feasible directions introduced next.

3 The Method of Feasible Directions

Let Ω be a feasible region of a general constrained optimization problem. Then a vector $\mathbf{d}$ is said to be a *feasible direction* at the point $\boldsymbol{\alpha} \in \Omega$, if there exists $\bar{\lambda}$ such that $\boldsymbol{\alpha} + \lambda\mathbf{d} \in \Omega$ for all $0 \leq \lambda \leq \bar{\lambda}$.

The main idea of the method of feasible direction is to find a path from the initial feasible solution to the optimal solution by making steps along feasible directions. At each iteration the feasible direction algorithm proceeds as follows:

- find the optimal feasible direction—that is, the feasible direction providing the largest rate of increase of the objective function.
- determine the step length along the feasible direction that maximizes the objective function ("line search").

The algorithm terminates when no feasible direction can be found which improves the objective function. For a general constrained optimization problem in the form

$$\begin{aligned} \text{Maximize} \quad & f(\boldsymbol{\alpha}) \\ \text{subject to:} \quad & A\boldsymbol{\alpha} \leq \mathbf{b} \end{aligned}$$

the optimal feasible direction is found by solving the direction finding linear program:

$$\begin{aligned} \text{Maximize} \quad & \nabla f^T \mathbf{d} \\ \text{subject to:} \quad & A\mathbf{d} \leq 0 \\ & ||\mathbf{d}|| \leq 1 \end{aligned} \tag{12}$$

The method of feasible directions can be applied directly to SVM training [8]. In this case, the respective optimal feasible direction problem can be stated as follows:

$$\begin{aligned} \text{Maximize} \quad & \tilde{\mathbf{g}}^T \mathbf{d} && (13) \\ \text{subject to:} \quad & \mathbf{c}^T \mathbf{d} = 0 && (14) \\ & d_i \geq 0, \quad \text{if } \alpha_i = 0 && (15) \\ & d_i \leq 0, \quad \text{if } \alpha_i = C && (16) \\ & ||\mathbf{d}|| \leq 1 && (17) \end{aligned}$$

where the definitions of $\tilde{\mathbf{g}}$ and $\mathbf{c}$ vary between the two SVM formulations.

It turns out that solving the linear problem of the full size at each iteration is expensive, and the overall performance of this method for SVM training is inferior to traditional optimization methods. However, with a slight modification, an approximate solution to the optimal feasible direction problem can be obtained in linear time, and this solution provides a powerful guidance for working set selection. This approximate solution lies at the core of the feasible direction decomposition algorithms described in the ensuing sections.

4 Feasible Direction Decomposition of the Pattern Recognition SVM

4.1 Joachim's Decomposition Algorithm

The key observation of Joachims is that adding a requirement that only q components of $\mathbf{d}$ be non-zero provides a straightforward working set selection rule: the variables corresponding to those non-zero components are to be included in the new working set. Unlike Zoutendijk's method, the optimal feasible direction vector is not to be followed exactly. With re-optimization assumed to be cheap, one can afford finding an optimal solution *in the entire subspace* spanned by non-zero components of $\mathbf{d}$, instead of doing line search strictly along $\mathbf{d}$. Unfortumately, with the additional constraint, the optimal feasible direction problem becomes intractable. Therefore, one has to seek an approximate solution. One such solution is realized in Joachims' algorithm. In the rest of this section a detailed account of this solution is presented—in order to provide some insight and to underline the ideas used in the extension of this algorithm to the regression SVM.

The approximate solution is obtained by changing the normalization constraint (17) to

$$d_i \in \{-1, 0, 1\}$$

With this normalization, in order to satisfy the equality constraint (14) (recall that for the pattern recognition SVM the equality constraint is $\mathbf{y}^T\mathbf{d} = 0$), it suffices that *the number of elements with sign matches between* d_i *and* y_i *is equal to the number of elements with sign mismatches between* d_i *and* y_i. Obviously, for a working set of size q this condition holds only when each number is equal to $q/2$. Therefore, the equality constraint (14) can be enforced by performing two passes on the data: the "forward pass" selects $q/2$ elements with sign mismatches, and the "backward pass"—$q/2$ elements with sign matches[3].

How should directions be determined for the elements? Recall that the goal is to maximize the objective function (13) subject to constraints. In the absense of constraints the maximum of the objective function would have been achieved by selecting q points with the highest values of $|g_i|$ and assigning directions $d_i = -\text{sign}(g_i)$ to them. Let's consider the largest contribution γ_k to the objective function provided by some point k, subject to equality constraints. During the forward pass the signs of d_k and y_k must be different, therefore:

$$\begin{aligned}
y_k = 1 \quad &\Rightarrow \quad d_k = -1 \Rightarrow \quad \gamma_k = \max_{i:\, y_i = 1} (-g_i) \Rightarrow \quad \gamma_k = \min_{i:\, y_i = 1} (g_i y_i) \\
y_k = -1 &\Rightarrow \quad d_k = 1 \quad \Rightarrow \quad \gamma_k = \max_{i:\, y_i = -1} (g_i) \Rightarrow \quad \gamma_k = \min_{i:\, y_i = -1} (g_i y_i)
\end{aligned}$$

and hence, combining the subscripts,

$$\gamma_k = \min_i (g_i y_i)$$

[3] The motivation for the names "forward" and "backward" will be clear shortly.

Likewise, for the backward pass the signs of d_k and y_k must be the same, therefore:

$$y_k = 1 \;\Rightarrow\; d_k = 1 \;\Rightarrow\; \gamma_k = \max_{i:\, y_i=1} (g_i) \;\Rightarrow\; \gamma_k = \max_{i:\, y_i=1} (g_i y_i)$$
$$y_k = -1 \Rightarrow\; d_k = -1 \Rightarrow\; \gamma_k = \max_{i:\, y_i=-1} (-g_i) \Rightarrow\; \gamma_k = \max_{i:\, y_i=-1} (g_i y_i)$$

and hence, combining the subscripts,

$$\gamma_k = \max_i (g_i y_i)$$

Thus the quantity $g_i y_i$ reflects the element's contribution to the objective function subject to the equality constraint. The working set composition rule can then be stated as follows: sort data elements by $g_i y_i$ in increasing order[4] and select $q/2$ elements from the front of the list (hence, the "forward pass"), and $q/2$ elements from the back of the list (hence, the "backward pass").

Finally, to account for inequality constraints (15) and (16), some points may have to be skipped if they violate one of these constraints. For such points, the direction leading to improvement of the objective function of the optimal feasible direction problem is infeasible.

Joachims' algorithm is summarized in Algorithm 1.

Algorithm 1 Joachims' SVM decomposition algorithm.

Let S be a list of all samples.
while (termination conditions (7) are not met)

- sort S by $y_i g_i$ in increasing order
- select $q/2$ samples from the front of S such that $0 < \alpha_i < C$ or $d_i = -y_i$ satisfies (15) and (16) (forward pass)
- select $q/2$ samples from the back of S such that $0 < \alpha_i < C$ or $d_i = y_i$ satisfies (15) and (16) (backward pass)
- re-optimize the working set

4.2 Optimality of a Working Set

As it was mentioned earlier, pointwise termination criteria in Osuna's and Joachims' algorithms require knowledge of the threshold b of the SVM. This threshold can be difficult to calculate, especially for the regression SVM. In this section an alternative termination condition is presented which allows to determine

[4] In practice, sorting which takes $O(n \log n)$ operations can be replaced with heap-based algorithms yielding the complexity of $O(n \log q)$ or $O(q \log n)$ depending on how the heap is built.

whether or not the entire working set is sub-optimal and hence suitable for re-optimization. The new conditions are based on examination of KKT conditioins for the standard form of a quadratic program. The exposition in this section concentrates on the pattern recognition SVM, whereas a similar result for the regression SVM is presented in section 5.1.

Consider the quadratic problem (1)–(2) of the pattern recongition SVM. The standard form of a quadratic program is obtained by transforming all constraints to either quality on non-negativity constraints, adding the slack variables when necessary. In particular, to cast the problem (1)–(2), a vector of slack variables $\mathbf{s}^T = (s_1, \ldots, s_l)$ is added for every $\alpha_i < C$ constraint, and a slack variable with the 0 value is added to represent the requirement for equality constraint to be satisified. For notational purposes the following matrices and vectors are introduced:

$$E = \begin{bmatrix} y_1 \; 1 \ldots 0 \\ \vdots \quad\; \ddots \\ y_l \; 0 \ldots 1 \end{bmatrix}, \quad E^* = \begin{bmatrix} E \\ I \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \boldsymbol{\alpha} \\ 0 \\ \mathbf{s} \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} 0 \\ C \\ \vdots \\ C \end{bmatrix} \tag{18}$$

In the matrix notation (18) all constraints of the original problem (1)–(2) can be compactly expressed as:

$$\begin{aligned} E^{*T}\mathbf{z} &= \mathbf{f} \\ \mathbf{z} &\geq \mathbf{0} \end{aligned} \tag{19}$$

Now the Karush-Kuhn-Tucker Theorem ([1], p. 36) can be stated as follows:

Theorem 1 (Karush-Kuhn-Tucker Theorem). *The primal vector* $\mathbf{z}$ *solves the quadratic problem (1) if and only if it satisfies (19) and there exists a dual vector* $\mathbf{u}^T = (\boldsymbol{\Pi}^T \; \mathbf{w}^T) = (\boldsymbol{\Pi}^T \; (\mu \; \boldsymbol{\Upsilon}^T))$ *such that:*

$$\boldsymbol{\Pi} = D\tilde{\boldsymbol{\alpha}} + E\mathbf{w} - \tilde{\mathbf{y}} \geq \mathbf{0} \tag{20}$$

$$\boldsymbol{\Upsilon} \geq \mathbf{0} \tag{21}$$

$$\mathbf{u}^T\mathbf{z} = 0 \tag{22}$$

It follows from the Karush-Kuhn-Tucker Theorem that if for all $\mathbf{u}$ satisfying conditions (21) – (22) the system of inequalities (20) is inconsistent then the solution of problem (1) is *not* optimal. Since the sub-problem (4) was obtained by merely re-arranging terms in the objective function and the constraints of the initial problem (1), the same conditions guarantee that the sub-problem (4) is not optimal. Thus, the main strategy for identifying sub-optimal working sets will be to enforce inconsistency of the system (20) while satisfying conditions (21) – (22).

Notice that all constant terms in (20) represent the negative gradient vector $-\mathbf{g} = D\boldsymbol{\alpha} - \tilde{\mathbf{y}}$. Thus each inequality in (20) can be written as follows:

$$\pi_i = -g_i + y_i\mu + \upsilon_i \geq 0 \tag{23}$$

Consider three cases, according to the values that α_i can take:

1. $0 < \alpha_i < C$. In this case, the complementarity condition (22) implies that $\pi_i = 0$, $\upsilon_i = 0$. Then inequality (23) becomes

$$-g_i + y_i\mu \;=\; 0 \tag{24}$$

2. $\alpha_i = 0$. In this case, the complementarity condition (22), implies that $\upsilon_i = 0$. Then inequality (23) becomes

$$-g_i + y_i\mu \;\geq\; 0 \tag{25}$$

3. $\alpha_i = C$. In this case, the complementarity condition (22), implies that $\pi_i = 0$. Then inequality (23) becomes

$$-g_i + y_i\mu \;\leq\; 0 \tag{26}$$

It can be easily seen that enforcing the complementarity constraint (22) causes μ to become the only free variable in the system (20). Each point restricts μ to a certain interval on a real line. Such intervals will be denoted as *μ-sets* in the rest of the article. The rules for computation of μ-sets can be summarized as follows:

$$\mathcal{M}_i = \begin{cases} [g_i/y_i], & \text{if } 0 < \alpha_i < C \\ [g_i, +\infty), & \text{if } \alpha_i = 0 \text{ and } y_i = 1 \\ (-\infty, -g_i], & \text{if } \alpha_i = 0 \text{ and } y_i = -1 \\ (-\infty, g_i], & \text{if } \alpha_i = C \text{ and } y_i = 1 \\ [-g_i, +\infty), & \text{if } \alpha_i = C \text{ and } y_i = -1 \end{cases} \tag{27}$$

The development in this section is summarized in the following theorem:

Theorem 2. *The vector $\boldsymbol{\alpha}$ solves the quadratic program (1) – (2) if and only if intersection of μ-sets computed by (27) is non-empty.*

It also follows from the expressions (27) that if at least one α_i is strictly between the bounds, then at the optimal solution the intersection of all μ-sets (which is non-empty by the Kuhn-Tucker theorem) is a single point set. This is consistent with the known property of SVM that at the optimal solution $b = \mu$. The intersection of μ-sets at the optimal solution can only be a non-empty, non single point set if all variables are at the bounds. In this case any point from the intersection of μ-sets can be taken as b; in particular, the value suggested in (11).

4.3 Maximal Inconsistency Algorithm

While inconsistency of the working set at each iteration guarantees convergence of decomposition, the rate of convergence is quite slow if arbitrary inconsistent working sets are chosen. A natural heuristic is to select "maximally inconsistent" working sets, in a hope that such choice would provide the greatest improvement of the objective function. The notion of "maximal inconsistency" is easy to define: let it be the gap between the smallest right boundary and the largest left boundary of μ-sets of elements in the training set:

$$G = L - R$$
$$L = \max_{0<i<l} \mu_i^l, \quad R = \min_{0<i<l} \mu_i^r$$

where μ_i^l, μ_i^r are the left and the right boundaries respectively (possibly minus or plus infinity) of the μ-set $\mathcal{M}_i$. It is convenient to require that the largest possible inconsistency gap be maintained between all pairs of points comprising the working set. The obvious implementation of this strategy is to select $q/2$ elements with the largest values of μ^l and $q/2$ elements with the smallest values of μ^r.

One feature of Joachims' method needs to be retained – rejection of Zoutendijk-infeasible points (cf. (15), (16)). Inclusion of such points into the working set doesn't make sense anyway because their values of α will not change after re-optimization. In the pattern recognition case, the μ-set constraints are not capable of encoding feasibility[5]. Since the notion of the direction is not explicitly maintained in the maximal inconsistency algorithm the feasibility test needs tobe modified slightly: the point is infeasible if $\alpha_i = 0$ and $g_i < 0$ or if $\alpha_i = C$ and $g_i < 0$.

The maximal inconsistency strategy is summarized in Algorithm 2.

Algorithm 2 Maximal inconsistency algorithm for the pattern recognition SVM.

Let S be a list of all samples.
while (intersection of all μ-sets is empty)

- compute $\mathcal{M}_i$ according to the rules (27) for all elements in S
- select $q/2$ feasible samples with the largest values of μ^l ("left pass")
- select $q/2$ feasible samples with the smallest values of μ^r ("right pass")
- re-optimize the working set

4.4 Equivalence between Joachims' Algorithm and the Maximal Inconsistency Algorithm

So far the motivation for the maximal inconsistency algorithm has been purely heuristic. Does inconsistency gap indeed provide a good measure of optimality

[5] They will be in the regression case.

for a working set at a given iteration? An affirmative answer is developed in this section, by showing the equivalence between Joachims' algorithm and the maximal inconsistency algorithm.

To show that the algorithms are equivalent we have to prove that they produce identical working sets at each iteration[6] and that their termination conditions are equivalent. The two propositions below handle each claim.

Proposition 1. *The working set of Joachims' algorithm is identical to the working set of maximal inconsistency algorithm at each iteration.*

Proof. The statement above will be proved for the half of the working set; namely, that the set of elements selected by the "forward pass" of Joachims' algorithm is identical to the set of elements selected by the "right pass" of the maximal inconsistency algorithm. Similar argument allows to establish equivalence of the other half of working sets.

Let F be the set of all feasible samples at some iteration. Let $r_J(p)$ denote the rank (position) of sample p in the array obtained by sorting F by $y_i g_i$; let $r_I(p)$ be the rank of sample p in the array obtained by sorting F by μ_i^r.

Let the set $H_J = \{p : r_J(p) \leq q/2\}$ be the half of the working set selected by the "forward pass" of Joachims' algorithm. Let $\bar{p}$ be the sample whose $r_J(\bar{p}) = q/2$, i.e. the element in H_J with the largest value of the key. Let $H_I = \{p : \mu^r(p) \leq \mu^r(\bar{p})\}$. We are going to prove that $H_J \equiv H_I$.

For any sample p_i selected by the "forward pass" $d_i = -y_i$. Considering the possible values of α_i, we conclude that, for any $p \in H_J$:

$$\begin{array}{lllll} 0 < \alpha_i < C & & & \Rightarrow & \mu_i^r = y_i g_i \\ \alpha_i = 0 & \Rightarrow d = 1 & \Rightarrow y_i = -1 & \Rightarrow & \mu_i^r = -g_i = -1 \cdot g_i = y_i g_i \\ \alpha_i = C & \Rightarrow d = 1 & \Rightarrow y_i = 1 & \Rightarrow & \mu_i^r = g_i = 1 \cdot g_i = y_i g_i \end{array} \tag{28}$$

We can see that the order of $\mu^r(p)$ is preserved when mapping $H_J \mapsto H_I$. Therefore, if $p \in H_J$ then $p \in H_I$.

To prove set equivalence, it remains to be shown that if $p \in H_I$ then $p \in H_J$. Suppose, by the way of contradiction this is not the case, i.e. there exists an sample $\tilde{p}$ such that $\mu^r(\tilde{p}) < \mu^r(\bar{p})$ and $\tilde{p} \notin H_J$. Since $\tilde{p} \notin H_J$, $r_J(\tilde{p}) \geq r_J(\bar{p})$. What is $\mu^r(\tilde{p})$ equal to? Three possible cases are covered in (28), and in each of them $\mu^r(\tilde{p}) = g_i y_i$. In this case $\mu^r(\tilde{p}) < \mu^r(\bar{p}) \Rightarrow r_J(\tilde{p}) < r_J(\bar{p})$ which contradicts our previous conclusion that $r_J(\tilde{p}) \geq r_J(\bar{p})$. In the remaining two cases of (27) $\mu^r(\tilde{p}) = +\infty$ which contradicts our assumption that $\mu^r(\tilde{p}) < \mu^r(\bar{p})$. Thus we can conclude that $H_J \equiv H_I$. □

Proposition 2. *Termination conditions of the Joachims' algorithm are satisfied if and only if the termination conditions of the maximal inconsistency algorithm are satisfied.*

[6] Since both algorithms use identical feasibility check for every sample, it is obvious that infeasible samples will never be included in a working set in both algorithms.

Proof. The maximal inconsistency algorithm terminates when the system (20) is consistent while at the same time conditions (21) – (22) are enforced. Hence, the KKT conditions are satisfied, and consequently, the algorithm terminates if and only if the optimal solution is found. Likewise, termination conditions (5) – (7) also have an "if and only if" relationship to the KKT conditions and hence to the optimality of the solution, except when the solution contains all variables strictly at the bounds and (11) is used for calculation of b. In the latter case, however, condition (11) satisfies the KKT conditions of the *primal* SVM training problem (to which the problem (1) – (2) is a dual). Then it follows from Dorn's duality theorem ([6]. p. 124), that the solution of the dual problem is also optimal. Hence, both algorithms terminate at the same point in the solution space. □

5 Feasible Direction Decomposition of the Regression SVM

5.1 Maximal Inconsistency Algorithm

We now turn our attention to the maximal inconsistency algorithm for the regression SVM. Recall that the quadratic program for the latter is given by equations (1) and (3). The derivation will progress in the same way as in the pattern recognition case and will consist of: (a) stating the Karush-Kuhn-Tucker Theorem for the standard form of QP, (b) derivation of rules for computation of μ-sets, and (c) defining the inconsistency gap to be used for working set selection in the algorithm.

The standard form of QP for the regression SVM is defined by the following matrices:

$$E = \begin{bmatrix} \mathbf{1} & I & \mathbf{0} \\ -\mathbf{1} & \mathbf{0} & I \end{bmatrix}, \quad \tilde{E} = \begin{bmatrix} E \\ I \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \tilde{\boldsymbol{\alpha}} \\ 0 \\ \mathbf{s} \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} 0 \\ \mathbf{C} \end{bmatrix} \tag{29}$$

in terms of which the constraints can expressed in the same way as in the pattern recognition case:

$$\begin{aligned} \tilde{E}^T\mathbf{z} &= \mathbf{f} \\ \mathbf{z} &\geq \mathbf{0} \end{aligned} \tag{30}$$

The statement of the Karush-Kuhn-Tucker Theorem, as well as its use to test optimality of the working set, remain the same (see Theorem 1 and the ensuing discussion).

For the regression SVM, each inequality in (20) has one of the following forms:

$$\pi_i = -\phi_i + \epsilon + \upsilon_i + \mu \geq 0 \tag{31}$$

$$\pi_i^* = \phi_i + \epsilon - \upsilon_i^* - \mu \geq 0 \tag{32}$$

where

$$\phi_i = y_i - \sum_{j=1}^{l} (\alpha_j - \alpha_j^*) K_{ij} \tag{33}$$

Considering the possible values of α_i we have:

1. $\alpha_i = 0$. In this case $s_i = C$, and, by complementarity condition (22), $\upsilon_i = 0$. Then inequality (31) becomes:

$$\pi_i \;=\; -\phi_i + \epsilon + \mu \;\geq\; 0 \quad \Rightarrow \quad \mu \;\geq\; \phi_i - \epsilon$$

2. $\alpha_i = C$. By complementarity condition (22), $\pi_i = 0$. Then inequality (31) becomes:

$$-\phi_i + \epsilon + \mu + \upsilon_i \;=\; 0 \quad \Rightarrow \quad \mu \;\leq\; \phi_i - \epsilon$$

3. $0 < \alpha_i < C$. By complementarity condition (22), $\upsilon_i = 0$, $\pi_i = 0$. Then inequality (31) becomes:

$$-\phi_i + \epsilon + \mu \;=\; 0 \quad \Rightarrow \quad \mu \;=\; \phi_i - \epsilon$$

Similar reasoning for α_i^* and inequality (32) yields the following results:

1. $\alpha_i^* = 0$. Then

$$\mu \;\leq\; \phi_i + \epsilon$$

2. $\alpha_i^* = C$. Then

$$\mu \;\geq\; \phi_i + \epsilon$$

3. $0 < \alpha_i^* < C$. Then

$$\mu \;=\; \phi_i + \epsilon$$

Finally, taking into account that for the regression SVM $\alpha_i \alpha_i^* = 0$, the rules for computation of μ-sets for the regression SVM are the following:

$$\mathcal{M}_i = \begin{cases} [\phi_i - \epsilon, \phi_i + \epsilon], & \text{if } \alpha_i = 0,\ \alpha_i^* = 0 \\ [\phi_i - \epsilon, \phi_i - \epsilon], & \text{if } 0 < \alpha_i < C,\ \alpha_i^* = 0 \\ (-\infty, \phi_i - \epsilon], & \text{if } \alpha_i = C,\ \alpha_i^* = 0 \\ [\phi_i + \epsilon, \phi_i + \epsilon], & \text{if } \alpha_i = 0,\ 0 < \alpha_i^* < C \\ [\phi_i + \epsilon, +\infty), & \text{if } \alpha_i = 0,\ \alpha_i^* = C \end{cases} \tag{34}$$

For the regression SVM, the new termination condition is stated in the following theorem.

Algorithm 3 Maximal inconsistency algorithm for the regression SVM.

Let S be a list of all samples.
while (intersection of μ-sets is empty)

- compute $\mathcal{M}_i$ according to the rules (34) for all elements in S
- select $q/2$ feasible samples with the largest values of μ^l ("left pass")
- select $q/2$ feasible samples with the smallest values of μ^r ("right pass")
- re-optimize the working set

Theorem 3. *The vector $\boldsymbol{\alpha}$ solves the quadratic program (1) – (3) if and only if intersection of μ-sets computed by (34) is non-empty.*

The maximal inconsistency algorithm for the regression SVM is summarized in Algorithm 3.

Feasibility of samples is tested by the following rule: during the left pass skip samples with $\alpha_i = C, \alpha^* = 0$; during the right pass skip samples with $\alpha_i = 0, \alpha^* = C$. Justification of this rule is given in Lemma 1 in Section 5.2.

5.2 Interpretation of the Maximal Inconsistency Algorithm in the Feasible Direction Framework

As it was shown in Section 4.4, the maximal inconsistency algorithm is equivalent to Joachims' algorithm which was motivated by Zoutendijk's feasible direction problem. In this section it will be demonstrated that the maximal inconsistency algorithm for the regression SVm can also be interpreted as a feasible direction algorithm.

Recall the SVM optimal feasible direction problem stated in (13) - (17). The problem-specific components $\tilde{\mathbf{g}}$ and $\mathbf{c}$ have the following expressions for the regression SVM:

$$\tilde{\boldsymbol{g}} = \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{g}^* \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{1} \\ -\mathbf{1} \end{bmatrix} \tag{35}$$

where

$$\begin{aligned} g_i &= \phi_i - \epsilon \\ g_i^* &= -\phi_i - \epsilon \end{aligned} \tag{36}$$

and ϕ is defined in (33). In addition, a feasible direction algorithm must satisfy the constraint

$$|\{d_i : d_i \neq 0\}| = q \tag{37}$$

To develop an equivalent feasible direction algorithm, we will construct a mapping $\Phi_{\tilde{\boldsymbol{\alpha}}}$ which maps the state of the maximal inconsistency algorithm to a direction vector $\mathbf{d}$, and a normalization $N_{\mathbf{d}}$ similar to Joachims' normalization which replaces (17). The construction will possess the following properties:

1. At each iteration, $\Phi_{\tilde{\alpha}}$ is the solution to the optimal feasible direction problem with normalization $N_{\mathbf{d}}$.
2. Termination condition of the maximal inconsistency algorithm holds if and only if the solution to the optimal feasible direction problem is a zero direction.

Intuitively, the first property shows that working sets selected by the maximal inconsistency algorithm are the same as those selected by a feasible direction algorithm using normalization $N_{\mathbf{d}}$. The second property ensures that both algorithms terminate at the same time.

Consider the normalization

$$d_i \in \{(1,0),(0,-1),(-1,0),(0,1),(0,0)\} \tag{38}$$

and the mapping

$$(\Phi_{\tilde{\alpha}})_i = \begin{cases} (1,0) \text{ or } (0,-1), \text{ whichever is feasible, during the left pass} \\ (-1,0) \text{ or } (0,1), \text{ whichever is feasible, during the right pass} \\ (0,0), \text{ if sample } i \text{ is infeasible or not reached} \end{cases} \tag{39}$$

For the sake of brevity, the optimal feasible direction problem comprising equations (13) - (16), (37), (38) will be denoted as the *feasible direction problem.*

First, we need to make sure that $\Phi_{\tilde{\alpha}}$ is not ambiguous, i. e. that only one of the non-zero directions suggested in (39) is feasible.

Lemma 1. *Mapping $\Phi_{\tilde{\alpha}}$ is not ambiguous.*

Proof. Let us denote directions $(1,0),(0,-1),(-1,0),(0,1)$ as Ia, Ib, IIa, IIb (Type I directions are assigned during the left pass, and Type II directions—during the right pass). Table 1 shows feasibility of different directions depending on the values of optimization variables: Infeasibility of directions marked by †

Table 1. Feasibility of directions

Optimization variables	Feasible	Infeasible
$\alpha_i = 0,\ \alpha_i^* = 0$	Ia, IIb	Ib, IIa
$0 < \alpha_i < C,\ \alpha_i^* = 0$	Ia, IIa	Ib, IIb†
$\alpha_i = C,\ \alpha_i^* = 0$	IIa	Ia, Ib, IIb†
$\alpha_i = 0,\ 0 < \alpha_i^* < C$	Ib, IIb	Ia†, IIa
$\alpha_i = 0,\ \alpha_i^* = C$	Ib	Ia†, IIa, IIb

is due to the special property of regression SVM that $\alpha_i \alpha_i^* = 0$. It follows from the table that at each pass only one direction is feasible. □

This lemma justifies the feasibility test of the maximal inconsistency algorithm. It is clear from Table 1 that during the left pass there is no feasible direction for samples with $\alpha_i = C$, $\alpha_i^* = 0$. Likewise, during the right pass there is no feasible direction for samples with $\alpha_i = 0$, $\alpha_i^* = C$.

The next two lemmas show that $\Phi_{\tilde{\boldsymbol{\alpha}}}$ provides a solution to the feasible direction problem.

Lemma 2. *$\Phi_{\tilde{\boldsymbol{\alpha}}}$ satisfies all constraints of the feasible direction problem.*

Proof. The equality constraint of the optimal feasible dirction problem for the regression SVM has the form:

$$\sum_{i=1}^{l} d_i = \sum_{i=1}^{l} d_i^* \tag{40}$$

Let m_1, m_2, n_1, n_2 be the number of selected elements with directions $\{1,0\}$,$\{-1,0\}$, $\{0,-1\}$, $\{0,1\}$ respectively. Then

$$\begin{aligned}
\sum_{i=1}^{l} d_i &= m_1 \cdot 1 + m_2 \cdot (-1) &&= m_1 - m_2 \\
\sum_{i=1}^{l} d_i^* &= n_1 \cdot (-1) + n_2 \cdot 1 &&= n_2 - n_1
\end{aligned}$$

By the selection policy:

$$m_1 + n_1 = q/2 = m_2 + n_2$$

from which it follows that

$$m_1 - m_2 = n_2 - n_1$$

Hence, $\Phi_{\tilde{\boldsymbol{\alpha}}}$ satisfies the equality constraint of the optimal feasible direction problem.

Inequality constraints and the cardinality constraint are trivially satisfied by the construction of $\Phi_{\tilde{\boldsymbol{\alpha}}}$. □

Lemma 3. *$\Phi_{\tilde{\boldsymbol{\alpha}}}$ provides the optimal value of the objective function of the feasible direction problem.*

Proof. Let B_l and B_r denote the halves of the working set selected by the left and the right passes respectively. Suppose, by the way of contradiction, that there exists a feasible sample i such that $g_i d_i > g_j d_j$ for some sample $j \in B_l \cup B_r$.

For any element k considered during the left pass

$$g_k d_k = \begin{cases} 1 \cdot g_k = \phi_k - \epsilon & = \mu_k^l \quad \text{if } (1,0) \text{ is feasible} \\ -1 \cdot g_k^* = -(-\phi_k - \epsilon) = \phi_k + \epsilon & = \mu_k^l \quad \text{if } (0,-1) \text{ is feasible} \end{cases}$$

Therefore, if $j \in B_l$, then

$$g_i d_i = \mu_i^l < \mu_j^l = g_j d_j$$

which contradicts the hypothesis that $g_i d_i > g_j d_j$.

Likewise, for any element k considered during the right pass

$$g_k d_k = \begin{cases} -1 \cdot g_k = -(\phi_k - \epsilon) & = -\mu_k^r \quad \text{if } (-1, 0) \text{ is feasible} \\ 1 \cdot g_k^* = -\phi_k - \epsilon = -(\phi_k + \epsilon) = -\mu_k^r & \text{if } (0, 1) \text{ is feasible} \end{cases}$$

Therefore, if $j \in B_r$, then

$$-g_i d_i = \mu_i^r > \mu_j^r = -g_j d_j$$

which contradicts the hypothesis that $g_i d_i > g_j d_j$. □

Lemma 4. *Intersection of μ-sets is non-empty if an only if the feasible direction problem has a zero solution.*

Proof. By Theorem 3, non-empty intersection of μ-sets implies optimality of the solution to the quadratic program (1) – (3) which rules out existence of a non-zero feasible direction which would otherwise have have led to a new optimal solution.

On the other hand, if the optimal solution to the feasible direction problem is zero, this implies that all other feasible directions have negative projections on the gradient vector, and hence decrease the value of the objective function. It follows that the solution of the quadratic program (1) – (3) is optimal, and hence intersection of μ-sets is non-empty. □

Lemmas 1 – 4 prove the two properties of the mapping $\Phi_{\tilde{\boldsymbol{\alpha}}}$ and the normalization $N_{\mathbf{d}}$ claimed earlier in this section.

6 Experimental Results

The aim of this section is to provide insight into some properties of feasible direction decomposition algorithms that might explain their behaviour in different situations. In particular, the following issues are addressed:

- Overall scaling factors. This is the traditional way of analyzing performance of SVM training algorithms, introduced by Platt [10] and Joachims [3]. To perform at least qualitative comparison with their results, a similar evaluation is performed for the maximal inconsistency algorithm.
- Experimental convergence rates. In traditional optimization literature algorithms are most often evaluated in terms of their convergence rates. Since the decomposition algorithms borrow their core ideas from optimization theory and are iterative, it is natural to attempt to establish their rates of convergence. It will be shown that the maximal inconsistency algorithm seems to have linear rate of convergence which is consistent with known linear convergence rates for gradient descent methods.

– Profiled scaling factors. While decreasing the number of iterations is highly desirable, it must not be achieved at a cost of significantly increasing the cost of an iteration. It is therefore important to investigate the "profile" of one iteration of the decomposition algorithm.

Experimental evaluation of the new algorithm was performed on the modified KDD Cup 1998 data set. The original data set is available under
http://www.ics.uci.edu/∼kdd/databases/kddcup98/kddcup98.html.
The following modifications were made to obtain a pure regression problem:

– All 75 character fields were eliminated.
– Numeric fields CONTROLN, ODATEDW, TCODE and DOB were elimitated.

The remaining 400 features and the labels were scaled between 0 and 1. Initial subsets of the training database of different sizes were selected for evaluation of the scaling properties of the new algorithm. Experiments were run on a SUN4U/400 Ultra-450 workstation with 300MHz clock and 2048M RAM. RBF kernel with $\gamma = 10$, termination accuracy[7] 0.001, and cache size of 300M were used. The value of the box constraint C was 1, and the working set of size 10 was used. Two sets of experiments were performed: one using the full set of 400 features, another one using only the first 50 features ("reduced set"). As it turns out, the second problem is more constrained, with a larger proportion of bounded support vectors. Also, for the full set of features kernel computation dominates the overall training time.

6.1 Overall Scaling Factors

The training times, with and without decomposition, for different samples sizes, are displayed in Tables 2 and 3, for the full and the reduced sets of features respectively. The scaling factors are computed by plotting training times versus sample sizes on a log-log scale and fitting straight lines. The SV-scaling factors are obtained in the same fashion, only using the number of unbounded support vectors instead of the sample size as an abscissa. The actual plots are shown in Figure 1.

It can be easily seen that decomposition improves the running time by an order of magnitude. It's scaling factors are also significantly better. The scaling factors are consistent with the scaling factors presented by Platt [10] and Joachims [3] for the pattern recognition SVM.

A number of other interesting findings can be made from the results above. First, it is easily seen that the training with decomposition does not produce an identical solution to training without. The solutions differ in the number of support vectors, especially for the more constrained problem with the reduced set of features. This difference is due to the fact that the termination conditions

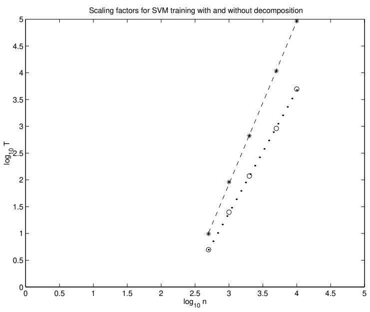
[7] In order for the results to be conceptually compatible with Joachims' I used the old point-wise termination conditions.

Table 2. Training time (sec) and number of SVs for the KDD Cup problem.

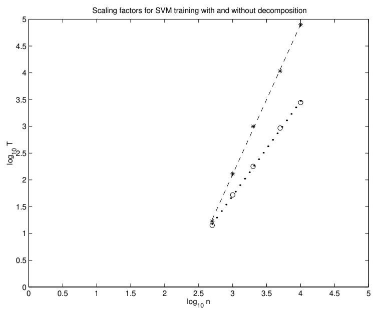
Examples	no dcmp			dcmp		
	time	total SV	BSV	time	total SV	BSV
500	9.818	184	0	4.962	170	0
1000	91.193	454	0	24.907	429	0
2000	665.566	932	2	118.131	894	2
5000	10785.4	2305	7	914.359	2213	7
10000	92684.7	4598	28	4958.26	4454	26
scaling factor:			3.03			2.29
SV-scaling factor:			2.86			2.13

Table 3. Training time (sec) and number of SVs for the KDD Cup problem, reduced feature set.

Examples	no dcmp			dcmp		
	time	total SV	BSV	time	total SV	BSV
500	16.958	114	29	14.175	128	26
1000	129.591	242	65	52.451	324	59
2000	998.191	445	114	178.752	656	104
5000	10759.6	977	323	929.523	1667	255
10000	79323.2	1750	633	2782.35	3383	491
scaling factor:			2.80			1.76
SV-scaling factor:			3.12			1.60



(a)



(b)

Fig. 1. Scaling factor fits: (a) full set of features (Table 2), (b) reduced set of features (Table 3)

of the decomposition algorithm require that KKT conditions be satisfied only to a given numerical precision. Thus the decomposition algorithm does have a disadvantage of producing an approximate solution, which can be further seen from Tables 4 and 5. These tables display the values of the objective functions attained by training with and without decomposition, and their ratio. The latter shows that in the relative turns, a solution of the more constrained problem with the reduced feature set is roughly 10 times worse that the solution of the less constrained problem with the fulls feature set. However, no deviation of the accuracy with the sample size is observed.

Another important observation is that the scaling factors and the SV-scaling factors vary among different problems. For the two particular problems, a possible explanation might be that the fixed termination accuracy is in fact looser for the more constrained problem, thereby producing a less accurate solution but taking less time in doing so. In general, however, the results above demonstrate that the scaling factors produce a rather crude measure of performance of the decomposition algorithms[8].

Table 4. Objective function values for KDD Cup problem.

Examples	no dcmp	dcmp	ratio
500	0.10799	0.10795	0.99962
1000	0.43501	0.43488	0.99970
2000	1.27695	1.27662	0.99974
5000	3.19561	3.19414	0.99953
10000	6.74862	6.74507	0.99947

Table 5. Objective function values for KDD Cup problem, reduced feature set.

Examples	no dcmp	dcmp	ratio
500	1.11129	1.10823	0.99724
1000	3.25665	3.24853	0.99750
2000	6.41472	6.39787	0.99456
5000	14.5490	14.5164	0.99776
10000	26.9715	26.9119	0.99779

[8] The scaling factors also vary in the results of Joachims [3].

6.2 Convergence Rates

In optimization literature, a common performance measure for iterative algorithms is convergence rate. The notion of the convergence rate is generally defined for any numeric sequence; for the purpose of analysis of decomposition algorithms we will be concerned with the sequence of objective function values. The following definitions are taken from [7].

Let x_k be a sequence in $\mathbb{R}^n$ that converges to x^*. Convergence is said to be *Q-linear* if there is a constant $r \in (0,1)$ such that

$$\frac{||x_{k+1} - x^*||}{||x_k - x^*||} \leq r, \quad \text{for all } k \text{ sufficiently large,}$$

The prefix "Q" stands for "quotient" because the quotient of successive distances from the limit point is considered. Likewise, convergence is said to be *Q-superlinear* if

$$\lim_{k \to \infty} \frac{||x_{k+1} - x^*||}{||x_k - x^*||} = 0,$$

and is said to be *of order p* (for $p > 1$) or *quadratic* (for $p = 2$) if

$$\frac{||x_{k+1} - x^*||}{||x_k - x^*||^p} \leq M, \quad \text{for all } k \text{ sufficiently large.}$$

where M is a positive constant not necessarily less than 1. It can be easily seen that a Q-convergent sequence of order strictly greater than 1 converges Q-superlinearly[9].

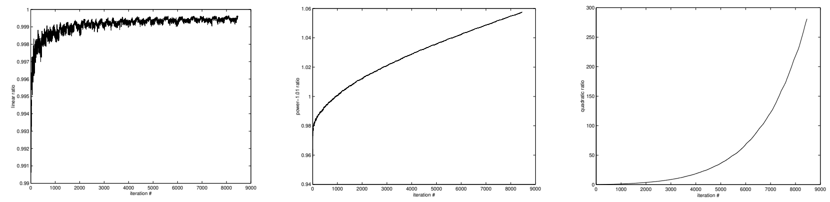
Convergence rates can be observed experimentally by recording the values of the objective function through the course if iteration and plotting the respective ratios versus the iteration number. Sample plots are shown in Figures 2 and 3 for both problems on a training sample of size 10000. Limit points have been obtained from training without decomposition. Similar plots have been observed in other experiments.

It is evident from the convergence plots that the decomposition algorithm converges linearly but not superlinearly or quadratically[10]. The plots also reveal that the training problem is very ill-conditioned, as the ratio stays very close to 1.

The above results are consistent with known linear convergence rates for gradient descent methods for unconstrained optimization. Noteworthy is particular semblance for the full feature set problem in which most of the variables stay away from the upper bounds and thus resemble an unconstrained case. Another important message is that convergence analysis (experimental or theoretical) is of special importance for decomposition algorithms. Unlike the scaling factors it reveals the effects of conditioning on the algorithm's performance.

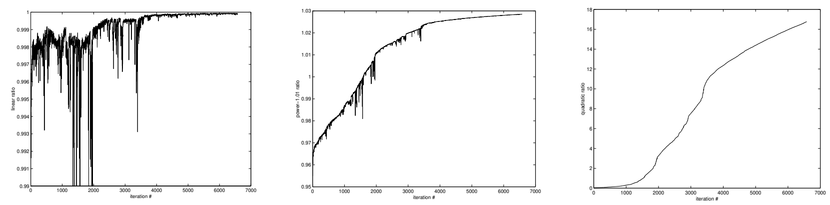
[9] The prefix "Q" will be omitted in the rest of the presentation in this section.

[10] It is difficult to see from the plot for the reduced feature set, but there is a tiny margin of ~ 0.0001 which seprates the ratios from 1.



(a) linear (b) $p = 1.01$ (c) quadratic

Fig. 2. Convergence rates for the full feature space



(a) linear (b) $p = 1.01$ (c) quadratic

Fig. 3. Convergence rates for the reduced feature space

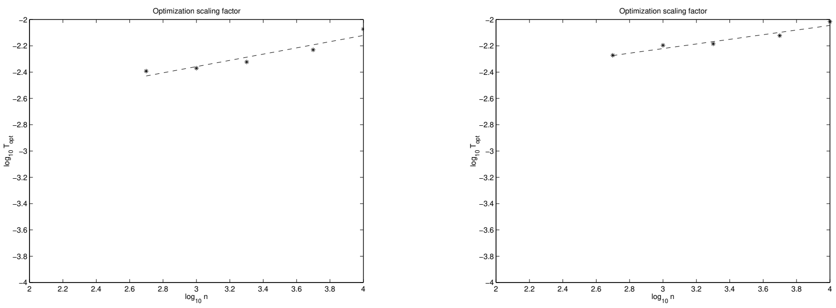
6.3 Profiled Scaling Factors

The overall processing performed during an iteration of the feasible direction decomposition algorithm can be broken up into 5 main steps:

1. Optimization: optimization proper and calculation of support vectors.
2. Update of gradients.
3. Kernel evaluation: all requests for dot products (from all modules of the system). Notice that with caching of kernel evaluations, this operation is not equally distributed across iterations: at the beginning it takes longer when kernels must be computed; towards the end all kernels end up in cache[11].
4. Selection: computation of maximal violation of KKT conditions, left and right passes.
5. Evaluation: computation of objective function and threshold.

In the following section the scaling factors per iteration are established for the five factors (overall scaling factors for kernel evaluation).

Optimization scaling factors. The values of obtained selection scaling factors are 0.237 for the full feature space and 0.176 for the reduced feature space, however the quality of fits is low. The expected behavior was constant-time per iteration, because the working sets are of constant size. Perhaps, for larger data sets conditioning of sub-problems deteriorates slightly, thus increasing the number of iterations in the optimizer[12]. The fits are displayed in Figure 4



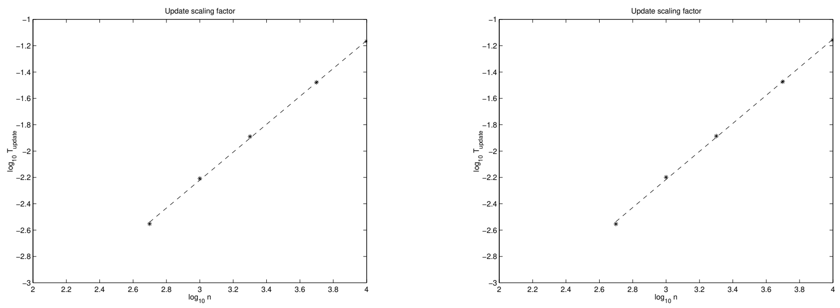
(a) full set of features (b) reduced set of features

Fig. 4. Optimization scaling factor fits.

[11] In the experiments above enough memory was allocated for the kernel cache to hold the values of kernels for all support vectors.

[12] The current implementation uses MINOS to solve optimization problems.

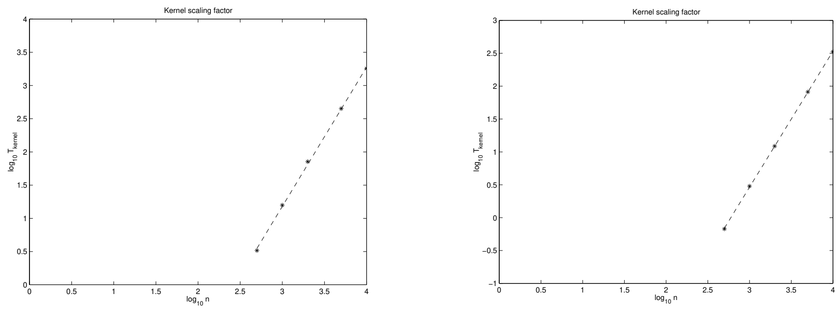
Update scaling factors. The values of obtained update scaling factors are 1.060 for the full feature space and 1.064 for the reduced feature space. This coinsides with the theoretical expectations of linear growth order of update operation. The fits are displayed in Figure 5.



(a) full set of features (b) reduced set of features

Fig. 5. Update scaling factor fits.

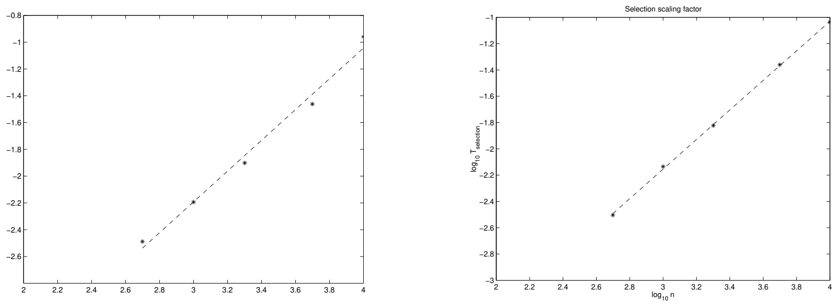
Kernel scaling factors. Kernel scaling factors are computed based on the timing accumulated over the entire run. The obtained values are 2.098 and 2.067 for the full and the reduced sets of features respectively. This coincides with the expected quadratic order of growth. The fits are displayed in Figure 6.



(a) full set of features (b) reduced set of features

Fig. 6. Kernel scaling factor fits.

Selection scaling factors. The values of obtained selection scaling factors are 1.149 for the full feature space and 1.124 for the reduced feature space. This is close to the theoretical expectations of linear growth order of selection operation[13]. The fits are displayed in Figure 7.



(a) full set of features

(b) reduced set of features

Fig. 7. Selection scaling factor fits.

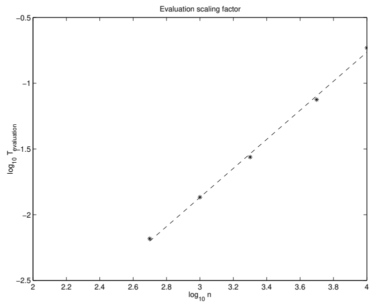
Evaluation scaling factors. The values of obtained evaluation scaling factors are 1.104 for the full feature space and 1.118 for the reduced feature space. This is close to the theoretical expectations of linear growth order of evaluation operation. The fits are displayed in Figure 8.

7 Conclusions

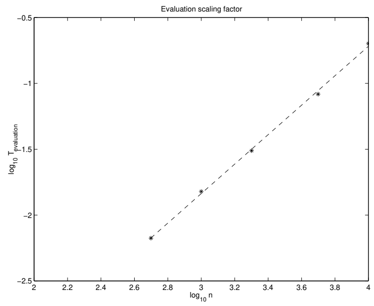
The unified treatment of the working set selection in decomposition algorithms presented in this article provides a general view of decomposition methods based on the principle of feasible direction, regardless of the particular SVM formulation. Implementation of the maximal inconsistency strategy is straightforward for both pattern recognition and regression SVM, and either of the termination conditions can be used. Formal justification of the maximal inconsistency strategy provides a useful insight into the mechanism of working set selection.

The experimental results demonstrate that, similar to the pattern recognition case, significant decrease of training time can be achieved by using the decomposition algorithm. While the scaling factors of the decomposition algorithms are significantly better than those of straightforward optimization, a word of

[13] The implementation uses heap-based method whose theoretical running time order is $O(n \log q)$. The logarithmic factor does not feature in the scaling factor because q is assumed constant.



(a) full set of features



(b) reduced set of features

Fig. 8. Evaluation scaling factor fits.

caution needs to be said with regard to the constants. It can be seen from the profiled experiments that the worst-case growth orders can only be given on a per-iteration basis, and the number of iterations, which depends on the convergence rate and the required precision, adds another dimension to running time analysis. The linear convergence rate observed in the experiments suggests that the progress towards the optimal solution can be slow. Additional investigation of the impact of problem conditioning is necessary.

A number of open questions remain regarding the SVM decomposition algorithms. Can the linear convergence rate be established theoretically? Can a super-linear or a quadratic convergence rate be achieved by a different algorithm? Finally, extremely far-reaching results can be borne by investigation of conditioning of the training problem: since the latter is a by-product of a number of factors, such as the choice of kernel, the box constraint, the risk functional, etc, conditioning of the optimization problem might be useful to guide the choice of SVM parameters.

References

1. John C. G. Boot. *Quadratic Programming.* North-Holland Publishing Company, 1964.
2. C.-C. Chang, C.-W. Hsu, and C.-J. Lin. The analysis of decomposition methods for support vector machines. In *Support Vector Machine Workshop, IJCAI*, 1999.
3. Thorsten Joachims. Making large-scale support vector machine learning practical. In Schölkopf et al. [12], pages 169–184.
4. Linda Kaufmann. Solving the quadratic problem arising in support vector classification. In Schölkopf et al. [12], pages 147–168.
5. Pavel Laskov. An improved decomposition algorithm for regression support vector machines. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 484–490. MIT Press, 2000.
6. Olvi L. Mangasarian. *Nonlinear Programming.* McGraw-Hill, 1969.

7. Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer-Verlag, 1999.
8. Edgar Osuna. *Support Vector Machines: Training and Applications.* PhD thesis, Massachusetts Institute of Technology, May 1998.
9. Edgar Osuna, Robert Freund, and Federico Girosi. Improved training algorithm for support vector machines. In *Neural Networks and Signal Processing*, 1997.
10. John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Schölkopf et al. [12], pages 185–208.
11. Bernhard Schölkopf. *Support Vector Learning.* PhD thesis, 1997.
12. Bernhard Schölkopf, Christopher Burges, and Alexander Smola, editors. *Advances in Kernel Methods – Support Vector Learning.* MIT Press, 1999.
13. Alexander Smola. *Learning with Kernels.* PhD thesis, Technical University of Berlin, 1998.
14. Alexander Smola and Bernhard Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2, 1998.
15. Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, New York, 1982.
16. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.
17. Vladimir N. Vapnik. *Statistical Learning Theory.* J. Wiley and Sons, 1999.
18. G. Zoutendijk. *Methods of Feasible Directions.* Elsewier, 1960.