Department of Electrical
and
Computer Systems Engineering

Technical Report
MECSE-26-2005

A Basic Heuristic Decomposition Framework for training Support Vector Machines

D. Lai, N. Mani and M. Palaniswami

MONASH
UNIVERSITY

A Basic Heuristic Decomposition Framework for training Support Vector Machines

D.Lai, N.Mani *Member, IEEE*, M.Palaniswami *Member, IEEE*

Abstract

Support Vector Machines are a form of binary classifier which requires optimization of a constrained quadratic program. When used to classify large data with limited processing memory, the main problem can be decomposed into smaller sub-problems which are then solved sequentially. However, the speed of this method depends on the sequence of sub-problems solved. In this paper, we propose a heuristic framework to select the sequence of sub-problems for faster convergence. The framework is based on the idea of maximizing the change in the objective function at each iteration. We provide a sufficient condition for any heuristic rule to guarantee asymptotic convergence of the decomposition method. The heuristics are then applied to benchmark data and show a consistent improvement in performance over current state of the art decomposition algorithms such as $\mathrm{SVM}^{\mathrm{light}}$ and LibSVM .

Index Terms

Support Vector Machines, classification, decomposition, inequality constraints, working set selection.

I. Introduction

The Support Vector Machines (SVM) developed by Vapnik [1] have been shown to be a powerful supervised learning tool for pattern recognition problems. The data to be classified is usually written as:

$$\begin{aligned} \mathbf{\Theta} &= \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ldots (\mathbf{x}_n, y_n)\} \\ \mathbf{x}_i &\in \Re^m \\ y_i &\in \{-1, 1\} \end{aligned} \tag{1}$$

The SVM formulation is essentially a regularized minimization problem leading to the use of Lagrangian Theory and quadratic programming techniques. The formulation defines a boundary separating two classes in the form of a linear hyperplane in data space where the distance between the boundaries of the two classes and the hyperplane is known as the margin. The idea is further extended for data that is not linearly separable by first mapping it (via a nonlinear function) to a possibly higher dimension feature space. The nonlinear function,

usually defined as $\phi(\mathbf{x}) : \mathbf{x} \subset \Re^n \rightarrow \Re^m, \;\; n << m$ is never explicitly used in the calculation. The Lagrangian dual problem expressed solely in terms of Lagrange multipliers, α_i namely:

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in \mathbf{D}} \Im(\boldsymbol{\alpha}) =& \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{G} \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{e} \\ \mathbf{D} =& \{\boldsymbol{\alpha} | 0 \leq \alpha_{\mathrm{i}} \leq C, \boldsymbol{\alpha}^T \mathbf{y} = 0\} \end{aligned} \tag{2}$$

is usually solved, where

$$G_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\mathbf{e} = \{1, 1, ..., 1\}$$

Explicit use of the nonlinear function $\phi(.)$, has been avoided through the use of a kernel function, defined formally as the dot products of the nonlinear functions

$$K(\mathbf{x}_i, \mathbf{x}_j) = < \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) > \tag{3}$$

We assume a Mercer kernel which is positive definite. The trained classifier then has the following form:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{4}$$

We will solve the following partitioned SVM problem via sequential decomposition using a projected Newton method

$$\begin{aligned} \min_{\boldsymbol{\alpha}_p} \max_{b} \Im\left(\boldsymbol{\alpha}'_p\right) =& \frac{1}{2} \begin{bmatrix} \boldsymbol{\alpha}'_p \\ \boldsymbol{\alpha}_s \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_p & \mathbf{H}_* \\ \mathbf{H}_*^T & \mathbf{G}_s \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}'_p \\ \boldsymbol{\alpha}_s \end{bmatrix} \\ & - \begin{bmatrix} \boldsymbol{\alpha}'_p \\ \boldsymbol{\alpha}_s \end{bmatrix}^T \begin{bmatrix} \mathbf{e}'_p \\ \mathbf{e}_s \end{bmatrix} \end{aligned} \tag{5}$$

subject to:

$$\mathbf{D}_p = \{\boldsymbol{\alpha}_p | \mathbf{0} \leq \boldsymbol{\alpha}_p \leq C\mathbf{1}\}$$

where $b \in \Re$, is now treated as a pseudo-Lagrangian multiplier [2]. The partitioned matrices are

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_p & \mathbf{G}_* \\ \mathbf{G}_*^T & \mathbf{G}_s \end{bmatrix} \quad \mathbf{H}_p = \begin{bmatrix} \mathbf{G}_p & \mathbf{y}_p \\ \mathbf{y}_p^T & 0 \end{bmatrix} \quad \mathbf{H}_* = \begin{bmatrix} \mathbf{G}_* \\ \mathbf{y}_s^T \end{bmatrix}$$

and the augmented vectors are

$$\boldsymbol{\alpha}'_p = \begin{bmatrix} \boldsymbol{\alpha}_p \\ b \end{bmatrix} \quad \mathbf{e}'_p = \begin{bmatrix} \mathbf{e}_p \\ 0 \end{bmatrix}$$

The vector $\boldsymbol{\alpha}'_p$ is known as the *working set* and $\mathbf{H}_p$ is the *subHessian* corresponding to the working set. Note that the gradient vector corresponding to the sub-problem is

$$\mathbf{v}'_p = \mathbf{H}_p\boldsymbol{\alpha}'_p + \mathbf{H}_*\boldsymbol{\alpha}_s - \mathbf{e}'_p \tag{6}$$

and the resulting Newton update is

$$\boldsymbol{\alpha'}_p^{t+1} = \boldsymbol{\alpha'}_p^{t} - \beta\mathbf{H}_p^{-1}\mathbf{v}'_p \tag{7}$$

where the *step magnitude* β is selected to ensure $\boldsymbol{\alpha'}_p^{t+1} \in \mathbf{D}$. Every element of the solution vector $\boldsymbol{\alpha}$ must satisfy the corresponding Karush-Kuhn-Tucker (KKT) conditions

$$y_i f(\mathbf{x}_i) \geq 1 \qquad if \quad \alpha_i = 0 \tag{8a}$$

$$y_i f(\mathbf{x}_i) \leq 1 \qquad if \quad \alpha_i = C \tag{8b}$$

$$y_i f(\mathbf{x}_i) = 1 \quad if \quad 0 < \alpha_i < C \tag{8c}$$

The decomposition method is generally applied in situations where computing memory is limited. The method decomposes the main problem into a series of sub-problems which are then solved sequentially where a sub-problem is identified by the corresponding set of variables known as *working sets* in SVM literature. Osuna [3] is credited as the earliest to apply a form of this method which he called *chunking* to solve face detection problems using SVMs. Later algorithms such as SMO [4] and $\mathrm{SVM}^{\mathrm{light}}$ [5] selected working sets based on approximately maximizing the step size. It has been recognized by Joachims [5], Lin and Hsu [6] and Chang [7] that the choice of working sets is central to the speed of the decomposition method. Lin [8], [9] showed that working sets chosen in the manner of $\mathrm{SVM}^{\mathrm{light}}$ resulted in a linear convergence rate. This was empirically confirmed by Laskov [10] who further showed that decomposition was sometimes faster than optimization on the entire problem space. The finite termination of SMO type algorithms has been proven by Keerthi and Gilbert [11] using a counting method. A general assumption is that the rate of convergence is proportional to the rate of improvement to the objective function [10]. In [12], we showed how the inequality constraints should be taken into account when selecting a working set in order to get better improvement to the objective function.

In this paper, we propose a heuristic framework for selecting working sets. The use of heuristics delivers a compromise between achieving faster theoretical convergence rates and efficient practical implementations. Our framework will be composed of semi-adaptive type heuristics which will be applied according to the properties of the dataset. In Section II,

we first begin by describing our notation and several well known results. Section III briefly outlines the working set problem and the use of the objective function as a merit function for choosing the working set. This is followed by some theoretical analysis on the projected Newton method in the decomposition setting. The exposition is necessary to motivate our design of the heuristic algorithm. We also derive a sufficient condition for a heuristic rule to guarantee asymptotic convergence of the heuristic algorithm. In Section V, we discuss the basic implementation of the framework and link our previous analysis to the described heuristics. We then run our proposed algorithm on several benchmark data sets and compare its performance against current state of the art algorithms i.e.$\mathrm{SVM}^{\mathrm{light}}$ and LibSVM. We use the standard notation where scalars are denoted by italics, column vectors by boldface small letters and matrices by boldface capitals.

II. Preliminaries and Notation

The *vector of variables* $\boldsymbol{\alpha}' \in \Re^{n+1}$ where n is the size of our data set $\boldsymbol{\Theta}$ is

$$\boldsymbol{\alpha}' = \{\alpha_1, ..., \alpha_n, b\}$$

The *working set*, $\boldsymbol{\alpha}'_p$ is a subset of $\boldsymbol{\alpha}'$ where

$$\boldsymbol{\alpha}'_p = \{\alpha_{p_1}, ..., \alpha_{p_k}, b\}$$

Throughout, we let the subscript *p* denote the selected working set while *s* indicates variables that do not change i.e. static during the optimization step. The subscripts also indicate the size of the vectors and the matrices e.g. if $\boldsymbol{\alpha}_p \in \Re^m$ then $\mathbf{G}_p \in \Re^{m \times m}$. The notation $\boldsymbol{\alpha}'_p$ denotes the augmented working set which includes the pseudo-Lagrangian *b*. Now let the set $\mathbb{A}$ denote the sigma algebra of working sets i.e. it contains all possible combinations of working sets so that $\forall p$ we have $\boldsymbol{\alpha}'_p \subseteq \mathbb{A}$. Since most of the state of the art decomposition algorithms e.g. SMO, LIBSVM, $\mathrm{SVM}^{\mathrm{light}}$ modify a small working set size we restrict our working set to the elements

$$\boldsymbol{\alpha}'_p = \{\alpha_1, \alpha_2, b\} \tag{9}$$

Let $\Delta z = z^{t+1} - z^t$ then (7) written element wise is

$$\Delta\alpha_1 = \beta \frac{y_1(y_2 v_2 - y_1 v_1 + (K_{21} - K_{22})v_b)}{\eta_{12}} \tag{10a}$$

$$\Delta\alpha_2 = \beta \frac{y_2(y_1 v_1 - y_2 v_2 + (K_{11} - K_{12})v_b)}{\eta_{12}} \tag{10b}$$

$$\Delta b = \beta\,[(K_{21} - K_{22})y_1 v_1 + (K_{12} - K_{11})y_2 v_2 + (K_{11}K_{22} - K_{12}K_{21})v_b] \tag{10c}$$

where due to the symmetric kernel function:

$$\eta_{12} = -\det \mathbf{H}_p = K_{11} + K_{22} - 2K_{12} \tag{11}$$

and $0 < \beta < 1$ is selected so that the updated working set satisfies the individual constraints i.e. $\boldsymbol{\alpha}'^{t+1}_p \in \mathbf{D}_p$. The corresponding element gradients are then

$$\mathbf{v}'_p = \{v_1, v_2, v_b\} \tag{12}$$

The error of a training example is defined as

$$E_i = y_i v_i \tag{13}$$

and during optimization, the violating points i.e. points that do not satisfy the KKT conditions, can be grouped according to the sign of the errors. The error indices sets are

$$\begin{aligned} I(\mathbf{E}^+) =& \{i | \alpha_i > 0, v_i > 0, y_i = 1\} \cup \\ & \{i | \alpha_i < C, v_i < 0, y_i = -1\} \\ I(\mathbf{E}^-) =& \{i | \alpha_i > 0, v_i > 0, y_i = -1\} \cup \\ & \{i | \alpha_i < C, v_i < 0, y_i = 1\} \end{aligned}$$

The general change in objective function [12] with respect to a working set $\boldsymbol{\alpha}'_p \subseteq \mathbb{A}$ is

$$M(\boldsymbol{\alpha}'_{\boldsymbol{p}}) = \frac{1}{2} \Delta \boldsymbol{\alpha}'^{\,T}_{\boldsymbol{p}} \mathbf{H}_p \Delta \boldsymbol{\alpha}'_{\boldsymbol{p}} + \Delta \boldsymbol{\alpha}'^{\,T}_{\boldsymbol{p}} \mathbf{v}'_p \tag{14}$$

The Frobenius norm or Euclidean matrix norm is defined for a $m \times n$ matrix $\mathbf{H}$ as

$$\|\mathbf{H}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |H_{ij}|^2}$$

When $n = 1$ this becomes the standard L_2-norm for vectors also known as the Euclidean vector norm. *Holder's inequality* (see for example [13]) states that for any $\mathbf{x}, \mathbf{y} \in \Re^n$

$$|\mathbf{x}^T \mathbf{y}| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q \quad \frac{1}{p} + \frac{1}{q} = 1$$

where the norms are *p*-norms defined as

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{\frac{1}{p}}$$

If the Euclidean norm is used, then the inequality is also known as the *Cauchy-Schwartz inequality*.

III. THE WORKING SET PROBLEM DEFINITION

The decomposition method presents an extra degree of freedom compared to standard Quadratic Programming programs e.g. LOQO [14], active set methods [15] in that one has the choice of selecting which working set to update at each iteration. In order to achieve respectable convergence rates, one generally requires a merit function to measure the optimality of the working set choice. A standard merit function employed in optimization is the change in objective function. The reason we would use this form of merit function is summarized by the following assumption

***Assumption** 3.1:* Let $\mathbf{T} = [0, t]$ denote the period of iteration and assume that in this period, **T** the rate of convergence is proportional to the change in the objective function.

In order words, we assume to get better convergence rates if we make the best progress possible towards the minimum of the objective function at every iteration. This written mathematically for every iteration step, t results in the following problem

$$\min_{\boldsymbol{\alpha}'_p \in \mathbb{A}} \frac{1}{2} \Delta \boldsymbol{\alpha}'^{T}_{\boldsymbol{p}} \mathbf{H}_p \Delta \boldsymbol{\alpha}'_{\boldsymbol{p}} + \Delta \boldsymbol{\alpha}'^{T}_{\boldsymbol{p}} \mathbf{v}'_p \tag{15}$$

where the *step size* $\Delta \boldsymbol{\alpha}'_{\boldsymbol{p}}$ is selected such that $\boldsymbol{\alpha}'^{t+1}_{p} \in \mathbf{D}_p$ and (15) is the change in the objective function for the SVM problem (2). This then becomes a combinatorial problem which needs to be solved at each iteration step until all KKT conditions or some criteria for finite termination is satisfied. Joachim's [5] bases his $\mathrm{SVM}^{\mathrm{light}}$ algorithm on solving

$$\min_{\boldsymbol{\alpha}'_p \in \mathbb{A}} \Delta \boldsymbol{\alpha}'^{T}_{\boldsymbol{p}} \mathbf{v}'_p \tag{16}$$

where again the step size $\Delta \boldsymbol{\alpha}'_p$ is selected to ensure feasibility. Problem (15) means searching for the working set which minimizes i.e. largest negative change in the objective function under Assumption 3.1. While solving (16) means finding the working set that gives largest change to $\boldsymbol{\alpha}'$ in the direction of steepest descent. Both methods are similar in nature. In our case, using the projected Newton method and substituting (7) reduces both (15) and (16) to the following problem

$$\min_{\boldsymbol{\alpha}'_p \in \mathbb{A}} -c_p \mathbf{v}'^{T}_{p} H_p^{-1} \mathbf{v}'_p \tag{17}$$

where c_p is some positive constant i.e. $c_p > 0$ to ensure feasibility of the updated working set. For our working set size, it will be shown later that (17) can be simplified further under certain initial conditions to

$$\min_{\boldsymbol{\alpha}'_p \in \mathbb{A}} \left(\frac{\beta^2}{2} - \beta \right) \frac{(E_2 - E_1)^2}{\eta_{12}} \tag{18}$$

From (15) it can be seen that the change in the objective function depends on the choice of the working set. Once the working set is selected, the magnitude of change in the objective function can be maximized by obtaining the largest step size $\Delta\boldsymbol{\alpha}'_p$. One may now conclude that the two immediate factors to improving the speed of the decomposition method are the choice of working set sequence and the update rule applied. However, neither the forward nor backwards Dynamic Programming methods [16] are suitable to be applied because all possible working set combinations have to be considered at each iteration. Instead of obtaining the optimal sequence of working sets, algorithms like SMO, $\mathrm{SVM}^{\mathrm{light}}$, LIBSVM and so on select working sets that approximately solve either (15) or (16). The smaller the number of iterations required, the better the theoretical convergence properties of the algorithm. Unfortunately, as we will demonstrate later, refining our search for the optimal working set at each iteration can compromise on the implementation run-time. We are thus interested in finding a compromise between these two opposing factors in designing an algorithm. The long term objective is to enhance the training times for Support Vector Machines so that they become more practical for large datasets. In the following, we propose the design of a basic heuristic framework with the objective of updating a shorter sequence of working sets as well as maintaining an acceptable if not better implementation run-time. The framework will be designed based on the view (see for example [17]) that specific problems require specialized techniques which take advantage of the problem structure, something that general optimization methods are unable to do.

In the following section, we provide some theoretical analysis to the minimization of (18). The results will be used to motivate the individual heuristics and the design of the heuristic framework. The initial fixation with the projected Newton rule coincides with the quadratic nature of the objective function and its quadratic convergence properties. We note however that the analysis can be extended easily to other update rules e.g. gradient descent, conjugate gradients etc.

IV. Theoretical Analysis of the Working Set problem

The purpose of this section is two-fold. First, we investigate the different components of the combinatorial problem (18) and establish some minor results. These are then used to compute the sensitivity of (18) with the aim of investigating which components play a bigger role. Then a sufficient condition is derived which needs to be satisfied by any heuristic rule to guarantee asymptotic convergence.

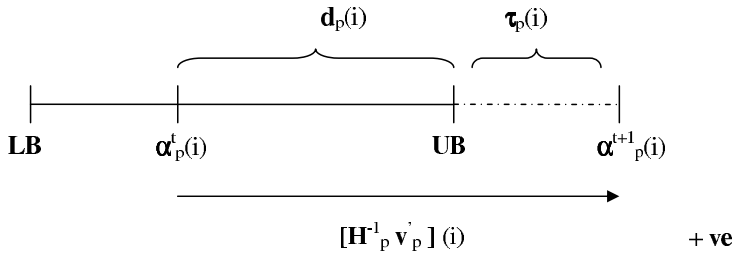

Fig. 1. The situation shown if the update step is positive and results in $\boldsymbol{\alpha}_p^{t+1}(i) > UB$. The potential overstep $\boldsymbol{\tau}_{p,i}$ and distance to bound $\mathbf{d}_{p,i}$ is positive.

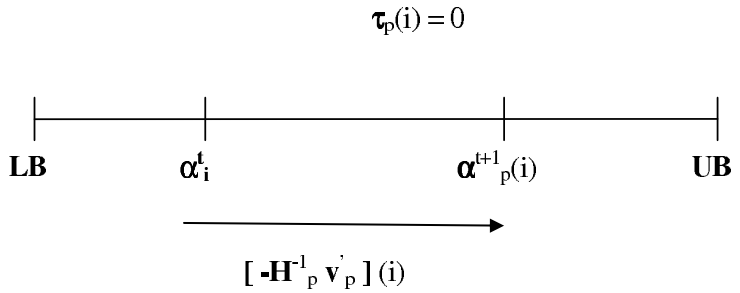

Fig. 2. The situation shown if the update step is $LB < \boldsymbol{\alpha}_p^{t+1}(i) < UB$. The potential overstep $\boldsymbol{\tau}_{p,i}$ is zero.

A. The effects of the inequality constraint set

Since this is a constrained quadratic problem, we expect the constraint set to affect the change in objective function. In line search algorithms, this has been implicitly done by adjusting the step magnitude β to maintain feasibility [15], [18], [19]. In order to study their effects, we proposed a method to quantify them. The structure of the inequality constraint set allows us to model their effect using a simple parameter $\boldsymbol{\tau}_{p,i}$ called the *potential of overstep* [12]. In relation to the Newton rule, this quantity models the amount of overstep if the update causes the i-th variable to violate the constraint bounds. Fig 1-Fig 2 illustrates the idea with respect to general upper (UB) and (LB) lower bounds. The distance to bounds $d_{p,i}$ measures the closeness of a variable to the bound. Mathematically, the two quantities are related by

$$\boldsymbol{\tau}_{p,i} = \begin{cases} -\mathbf{H}^{-1}_{p,i}\mathbf{v}'_p - d_{p,i} & \alpha_i^{t+1} > UB, \alpha_i^{t+1} < LB \\ 0 & \mathrm{LB} \leq \alpha_i^{t+1} \leq \mathrm{UB} \end{cases}$$

The augmented vector of potential overstep can be written as

$$\boldsymbol{\tau}'_p = -\mathbf{H}^{-1}_p \mathbf{v}'_p - \boldsymbol{d}'_p \tag{19}$$

where $\boldsymbol{d}'_p = \{\boldsymbol{d}_p, d_b\}$. The scalar $d_b = 0$ is for mathematical convenience since b is treated as a working set variable. Note that b is unbounded and so the notion of overstepping does not

apply. The constrained Newton update (7) rewritten in terms of the potential of overstep is then

$$\boldsymbol{\alpha'}_p^{t+1} = \boldsymbol{\alpha'}_p^{t} - \mathbf{H}_p^{-1}\mathbf{v}'_p - \boldsymbol{\tau}'_p \tag{20}$$

which ensures that $\boldsymbol{\alpha}_p^{t+1} \in \mathbf{D}_p$ minimizes the corresponding term in (5).

For our case of applying a projected Newton step, we project the updated iterate back into the feasible region by scaling the step size $\Delta\boldsymbol{\alpha}'_p$ so that only one multiplier is allowed to reach the bounds. Then the step magnitude β can be replaced as follows

$$\begin{aligned} -\beta\mathbf{H}_p^{-1}\mathbf{v}'_p &= -\mathbf{H}_p^{-1}\mathbf{v}'_p - \boldsymbol{\tau}'_p \\ \boldsymbol{\tau}'_p &= (\beta-1)\mathbf{H}_p^{-1}\mathbf{v}'_p \end{aligned} \tag{21}$$

We define the sets

$$\begin{aligned} A &= \left\{ \frac{C-\alpha_i^t}{\Delta\alpha_i} | i \in I(\boldsymbol{\alpha}_p), \alpha_i^{t+1} > UB \right\} \\ B &= \left\{ -\frac{\alpha_i^t}{\Delta\alpha_i} | i \in I(\boldsymbol{\alpha}_p), \alpha_i^{t+1} < LB \right\} \end{aligned}$$

Let $a = \min A_i \in A$ and $b = \min B_i \in B$. The scaled distance to bounds is now

$$d_{p,i} = -\min\{a,b\}\,\mathbf{H}_{p,i}^{-1}\mathbf{v}'_p$$

One may also notice that $\beta = \min\{a,b\}$ and we have thus described how to obtain β.

The following minor result shows that under a certain initial condition, the gradient with respect to the pseudo Lagrangian b remains static.

Lemma 4.1: Suppose that we select the working set $\boldsymbol{\alpha}'_p$ such that

$$I(\{\alpha_{i_1}, \alpha_{i_2}, b\}) = \{i_1, i_2, b | i_1, i_2 \in I(\mathbf{E}), b \in \Re\} \tag{22}$$

where $\mathbf{E} = \{y_i v_i\}$ for all $i = 1 \ldots n$. Assume for $t = 0$, $\alpha_i = 0$ for all $i = 1 \ldots n$. If for any $\boldsymbol{\alpha}'_p$ we apply (10a) to (10c) then for all $t \geq 0$ we have $v_b = 0$. Furthermore, if for some $t = t_k$, $v_b = 0$ then for all $t > t_k$ we have $v_b = 0$.

Proof: First, by differentiating (5) with respect to the threshold b, we have

$$v_b = y_{i_1}\alpha_{i_1} + y_{i_2}\alpha_{i_2}$$

From initial conditions, the lemma is proved trivially for $t = 0$. Now for $t > 0$, the change in the gradient is linear and can be written simply as

$$\Delta v_b = v_b^{t+1} - v_b^t = y_{i_1}\Delta\alpha_{i_1} + y_{i_2}\Delta\alpha_{i_2}$$

Substituting with (10a) and (10b) we obtain

$$\begin{aligned}\Delta v_b =& y_{i_1}\Delta\alpha_{i_1} + y_{i_2}\Delta\alpha_{i_2} \\ =& \beta\frac{y_{i_1}^2(y_{i_2}v_{i_2} - y_{i_1}v_{i_1} + (K_{i_2i_1} - K_{i_2i_2})v_b)}{\eta_{i_1i_2}} \\ &+\beta\frac{y_{i_2}^2(y_{i_1}v_{i_1} - y_{i_2}v_{i_2} + (K_{i_1i_1} - K_{i_2i_1})v_b)}{\eta_{i_1i_2}} \\ =& \frac{\beta(K_{i_1i_1} - K_{i_2i_2})v_b}{\eta_{i_1i_2}}\end{aligned}$$

Since $v_b = 0$ at $t = 0$, then $\Delta v_b = 0$ for $t = 1$ and $v_b^1 = 0$. Then by induction, we have $v_b^t = 0$ for all $t > 0$. Now since t is arbitrary, let $t = t' - t_k$ and the lemma holds also for $t' = t_k$. ■

The following result shows that under a certain selection rule, the elements which are likely to have $\boldsymbol{\tau}_p' > 0$ can be identified.

Lemma 4.2: Let the variables of a working set be denoted by $\boldsymbol{\alpha}_p' = \{\alpha_{i_1}, \alpha_{i_2}, b\}$ and assume we use a positive definite kernel with initial condition $\boldsymbol{\alpha} = \mathbf{0}$. Suppose that we select the working set such that

$$I(\boldsymbol{\alpha}_p') = \{i_1, i_2, b | i_1 \in I(\mathbf{E}^+), i_2 \in I(\mathbf{E}^-)\} \tag{23}$$

and the update rule of (10a)-(10c) then we have

a) If $v_i < 0$ and $\alpha_i^t = 0$ then $\alpha_i^{t+1} \rightarrow UB$.

b) If $v_i > 0$ and $\alpha_i^t = C$ then $\alpha_i^{t+1} \rightarrow LB$.

c) If $0 < \alpha_i^t < C$ and $v_i > 0$ then $\alpha_i^{t+1} \rightarrow LB$ while if $v_i < 0$ then $\alpha_i^{t+1} \rightarrow UB$

where motion of violaters when updated in relation to the bounds is denoted by "$\rightarrow$".

Proof: First note that the pseudo Lagrangian b is an unbounded variable and hence is not considered here. By Lemma 4.1, the initial condition of $\boldsymbol{\alpha} = \mathbf{0}$ implies $v_b = 0$ always[1] which means that the updates can be simplified. Now consider the element-wise updates written in the following manner

$$\Delta\alpha_{i_1} = y_{i_1}d \tag{24a}$$

$$\Delta\alpha_{i_2} = -y_{i_2}y_{i_1}\Delta\alpha_{i_1} = -y_{i_2}d \tag{24b}$$

where $d = \frac{(E_{i_2} - E_{i_1})}{\eta_{12}}$ and $\Delta\boldsymbol{\alpha}_p' = \{\Delta\alpha_{i_1}, \Delta\alpha_{i_2}, \Delta b\}$. Note that due to (23) we have $d \leq 0$ always since by definition of positive definiteness of the kernel we have $\eta_{12} > 0$. We consider the following cases:

[1]A proof for the general working set case can be found in [2].

1) Case $\alpha_{i_1} = 0$ and $v_{i_1} < 0$:

   From (8) this is a violater and $y_{i_1} = -1$ since $y_{i_1} v_{i_1} = E_{i_1} \in \mathbf{E}^+$. Provided the step is nonzero, from (24a) we have $\Delta\alpha_{i_1} > 0$ which implies $\Delta\alpha_{i_1} v_{i_1} < 0$. Furthermore, by definition $\Delta\alpha_{i_1} \geq 0$ means that $\boldsymbol{\alpha}_p^{t+1} > \boldsymbol{\alpha}_p^t$ or $\boldsymbol{\alpha}_p^t$ moves to the upper bound when updated.

2) Case $\alpha_{i_1} = C$ and $v_{i_1} > 0$

   The point is also a violater with $y_{i_1} = 1$ since by assumption (23) we have $y_{i_1} v_{i_1} = E_{i_1} \in \mathbf{E}^+$. Using (24a) as before we get $\Delta\alpha_{i_1} < 0$ and deduce $\Delta\alpha_{i_1} v_{i_1} < 0$.

3) Case $0 < \alpha_{i_1} < C$ and $v_{i_1} \neq 0$:

   This gives rise to similar situations as in Case 1 and Case 2 and the same result follows.

The proof for α_{i_2} follows in similar fashion using (24b) and is omitted here. ■

Corollary *4.1:* Assume a positive definite kernel, then for every violater, α_i where $i \in I(\mathbf{E}^+ \cup \mathbf{E}^-)$ updated using (7), the following holds

$$\Delta\alpha_i v_i < 0 \tag{25}$$

Proof: The proof follows easily by reexamining Cases 1-3 in Lemma 4.2 and noting that for all types of violaters, the product (25) is always negative. ■

The corollary is not surprising as it turns out to be a special case of Wolfe's theorem (see e.g. [19]) which we will invoke later on. In other words, working sets composed of violaters provide a strict decrease in the objective function. The next result requires the following

Lemma 4.3: Let the variables of a working set be denoted by $\boldsymbol{\alpha}'_p = \{\alpha_1, \alpha_2, b\}$ and assume a positive definite kernel matrix with initial condition $\boldsymbol{\alpha} = \mathbf{0}$. Then provided the step $\Delta\boldsymbol{\alpha}'_p \neq \mathbf{0}$ we have

$$\mathbf{v}'^T_p \mathbf{H}_p^{-1} \mathbf{v}'_p \geq 0 \tag{26}$$

with equality if and only if $\|\mathbf{v}'_p\| = 0$.

Proof: The proof is through direct calculation. First note by (7) that

$$\mathbf{H}_p^{-1} \mathbf{v}'_p = -\frac{1}{\beta} \Delta\boldsymbol{\alpha}'_p$$

Now using the initial conditions, we have from Lemma 4.1 that $v_b = 0$. Using (10a)-(10c)

we can thus simplify the following

$$\begin{aligned}
&{\mathbf{v}'_p}^T \mathbf{H}_p^{-1} \mathbf{v}'_p \\
&= -\frac{1}{\beta} {\mathbf{v}'_p}^T \Delta\boldsymbol{\alpha}'_p \\
&= -\frac{y_1 v_1 (y_2 v_2 - y_1 v_1)}{\eta_{12}} - \frac{y_2 v_2 (y_1 v_1 - y_2 v_2)}{\eta_{12}} \\
&= \frac{E_1^2 + E_2^2 - 2E_1 E_2}{\eta_{12}} = \frac{(E_2 - E_1)^2}{\eta_{12}} \geq 0
\end{aligned} \tag{27}$$

Note that (27) can be zero only if $v_1 = v_2 = 0$ or $\|\mathbf{v}'_p\| = 0$. The case $y_1 v_1 = y_2 v_2$ does not occur because $\Delta\boldsymbol{\alpha}'_p \neq \mathbf{0}$. ■

The following describes several properties of the potential of overstep and its relationship to the original step magnitude, β.

Lemma 4.4: Assume that $\lim_{t\to\infty} M(\boldsymbol{\alpha}') = 0$ and apply the Newton method (7) to solve (2), then $\forall \boldsymbol{\alpha}'_p \subseteq \mathbb{A}$ and iteration steps, $t > 0$ we have

a) $M(\boldsymbol{\alpha}'_{\boldsymbol{p}}) < 0$ if and only if $0 < \beta < 2$.

b) If in addition to $\lim_{t\to\infty} M(\boldsymbol{\alpha}') = 0$, for all $t > 0$ $\mathbf{v}'_p \neq \mathbf{0}$ then $\lim_{t\to\infty} \beta = 0$ or $\lim_{t\to\infty} \beta = 2$.

c) The condition $\lim_{t\to\infty} M(\boldsymbol{\alpha}') = 0$ further implies $\lim_{t\to\infty} \frac{\|\boldsymbol{\tau}'_p\|}{\|\mathbf{H}_p^{-1}\|\|\mathbf{v}'_p\|} \leq 1$ and the behaviour is *p*-norm independent.

Proof: First note that the assumption $\lim_{t\to\infty} M(\boldsymbol{\alpha}') = 0$ implies that in the limit, we have also for all $\boldsymbol{\alpha}'_p \subseteq \mathbb{A}$ the behaviour $\lim_{t\to\infty} M(\boldsymbol{\alpha}'_p) = 0$. Now set $\Delta\boldsymbol{\alpha}'_p = -\beta \mathbf{H}_p^{-1} \mathbf{v}'_p$ and substituting into (14) we obtain

$$M(\boldsymbol{\alpha}'_{\boldsymbol{p}}) = \left(\frac{\beta^2}{2} - \beta\right) {\mathbf{v}'_p}^T \mathbf{H}_p^{-1} \mathbf{v}'_p \tag{28}$$

For the Newton step to be a direction of descent, we must have

$$M(\boldsymbol{\alpha}'_{\boldsymbol{p}}) = \left(\frac{\beta^2}{2} - \beta\right) {\mathbf{v}'_p}^T \mathbf{H}_p^{-1} \mathbf{v}'_p < 0$$

From Lemma 4.3, we have ${\mathbf{v}'_p}^T \mathbf{H}_p^{-1} \mathbf{v}'_p > 0$ and so we require $\left(\frac{\beta^2}{2} - \beta\right) < 0$. The result of a) then follows after rearranging the inequality. The "only if" can be verified by taking any $\beta > 2$ or $\beta < 0$ to show that $M(\boldsymbol{\alpha}'_{\boldsymbol{p}}) > 0$. Taking limits with respect to t, we have

$$\lim_{t\to\infty} M(\boldsymbol{\alpha}'_{\boldsymbol{p}}) = \lim_{t\to\infty} \left(\frac{\beta^2}{2} - \beta\right) {\mathbf{v}'_p}^T \mathbf{H}_p^{-1} \mathbf{v}'_p$$

Now if we have $\lim_{t\to\infty} M(\boldsymbol{\alpha}'_p) = 0$ and $\mathbf{v}'_p \neq \mathbf{0}$ then we must have

$$\lim_{t\to\infty} \left(\frac{\beta^2}{2} - \beta\right) = 0 \tag{29}$$

which occurs if either $\lim_{t\to\infty}\beta = 0$ or $\lim_{t\to\infty}\beta = 2$. Taking norms of (21), we obtain

$$\begin{aligned} \|\boldsymbol{\tau}'_p\| &\leq |\beta - 1|\|\mathbf{H}_p^{-1}\|\|\mathbf{v}'_p\| \\ \rightarrow |\beta - 1| &\geq \frac{\|\boldsymbol{\tau}'_p\|}{\|\mathbf{H}_p^{-1}\|\|\mathbf{v}'_p\|} \end{aligned} \tag{30}$$

Note this is holds for any *p*-norm and so the result will be p-norm independent. From b) we deduce the following

$$\lim_{t\to\infty}|\beta - 1| = |\lim_{t\to\infty}\beta - 1| = 1$$

Using this and taking limits of (30)

$$\lim_{t\to\infty}|\beta - 1| \geq \lim_{t\to\infty}\frac{\|\boldsymbol{\tau}'_p\|}{\|\mathbf{H}_p^{-1}\|\|\mathbf{v}'_p\|} \tag{31}$$

and the result of c) follows. ■

The result in c) has several implications. As $t \rightarrow \infty$ or as we approach asymptotic convergence, multipliers at both upper and lower bounds will have non-zero potential of overstepping. The magnitude of overstepping increases indicating that they ”push” harder against the constraint boundaries. Multipliers in between the bounds will have some potential of overstep. Those in between bounds with zero gradients will not have a potential of overstep i.e. $\|\boldsymbol{\tau}'_p\| = 0$.

***Remark** 4.1:* A previous view was that a larger step size could be obtained if we minimized the potential of overstep [20]. A more appropriate requirement is to say that we want to find the largest step in the constrained region. Minimization of the potential of overstep not only ensures the step remains in the constrained region but implicitly looks for the largest step size.

B. The gradient elements and the condition of the subHessian

First, since we modify two multipliers and the threshold *b*, we can approximate the maximum gradient norm by

$$\|\mathbf{v}_p\|_{max} \approx |E_i - E_j|_{max}$$

where $\forall i, j$ we select $E_i \in \mathbf{E}^+, E_j \in \mathbf{E}^-$ and $v_b = 0$ from initial conditions. This selection method can be relaxed in the event that $\mathbf{E}^+ = \emptyset$ or $\mathbf{E}^- = \emptyset$ and further facilitated if the sets $\mathbf{E}^+$ and $\mathbf{E}^-$ are ordered as

$$\mathbf{E}^+ = \{E_i > 0 | E_1 \geq E_2 \geq E_3 \geq \ldots\} \tag{32a}$$

$$\mathbf{E}^- = \{E_i < 0 | E_1 \leq E_2 \leq E_3 \leq \ldots\} \tag{32b}$$

and one can pick in subsequent order the elements which would give maximum gradient norm. If either one of the sets is empty, one can still pick the first and last elements of the nonempty set to give a maximum gradient norm. This in fact is not new and has been the guiding heuristic in previous algorithms where the maximal violating pair has been chosen e.g. SMO,$\mathrm{SVM}^{\mathrm{light}}$ and LibSVM.

The matrix $\mathbf{H}_p$ which corresponds to the working set $\boldsymbol{\alpha'}_p$ is the Hessian of the subproblem (5). The condition number of this subHessian is defined as

$$\kappa_{\mathbf{H}_p} = \|\mathbf{H}_p\|\|\mathbf{H}_p^{-1}\|$$

and is a rough indication of the closeness to singularity of the matrix [13]. The smaller the condition number, the better conditioned the matrix is. Further more, the subHessian $\mathbf{H}_p$ defines the curvature of the subproblem (5) and a stronger subHessian e.g. diagonal dominant, means comparatively steeper directions of descent. Calculating the condition number or determining the diagonal dominance of the subHessian might be too computational intensive considering the combinations required.

We note that the quantity η_{12} defined in (11) is the determinant of $\mathbf{H}_p$ and explicitly appears in (18). Minimizing η_{12} can also be seen as approximately obtaining a larger change in the objective function which invariably means a steeper descent direction. It is an approximation mainly because a small determinant does not necessarily mean a more diagonally dominant matrix. However, for our heuristic purposes we could use this as a guide to select a subproblem with a relatively better conditioned subHessian.

C. Sensitivity Analysis of the Objective Function

In this part, we investigate the sensitivity of the discussed quantities in an attempt to pinpoint those which would contribute more towards the improvement of the objective. By substitution of (20) in (14) the change in the objective (2) can be written as

$$M(\boldsymbol{\alpha}'_p) = \frac{1}{2}\left[\boldsymbol{\tau'}_p^T \mathbf{H}_p \boldsymbol{\tau'}_p - \mathbf{v}_p'^T \mathbf{H}_p^{-1} \mathbf{v}'_p\right] \tag{33}$$

where the Hermitian matrix $\mathbf{H}_p$ is the Hessian of the sub-problem, $\mathbf{v}'_p$ is the augmented gradient vector and $\boldsymbol{\tau'}_p$ is the vector of potential overstep. The following technical lemma will be used for the next result.

Lemma 4.5: Let $\mathbf{H} \in \Re^{n\times n}$ be nonsingular and $\| \bullet \|$ be any matrix norm. Then $\mathbf{H} + \Delta\mathbf{H}$ is nonsingular provided

$$\|\Delta\mathbf{H}\| < \frac{1}{\|\mathbf{H}^{-1}\|} \tag{34}$$

and the inverse expansion exists and is

$$(\mathbf{H}+\Delta\mathbf{H})^{-1} = \mathbf{H}^{-1} - (\mathbf{H}^{-1}\Delta\mathbf{H})\mathbf{H}^{-1} + (\mathbf{H}^{-1}\Delta\mathbf{H})^2\mathbf{H}^{-1} - \ldots \quad (35)$$

Proof: see for example Adi-Ben [21] ■

The matrix $\Delta\mathbf{H}$ can then be viewed as a perturbation of $\mathbf{H}$. We propose the following

Lemma 4.6: Let $\mathbf{H} \in \Re^{n\times n}$ be positive definite and let $\| \bullet \|$ denote the Frobenius matrix norm. For any positive definite $\Delta\mathbf{H} \in \Re^{n\times n}$ such that

$$\|\Delta\mathbf{H}\| < \frac{1}{2\|\mathbf{H}^{-1}\|} \quad (36)$$

and $\mathbf{x} \in \Re^n$ we have

$$\mathbf{x}^T(\mathbf{H}^{-1} - \Delta\mathbf{H}^{-1})\mathbf{x} \leq \mathbf{x}^T(\mathbf{H}-\Delta\mathbf{H})^{-1}\mathbf{x} \leq \mathbf{x}^T(\mathbf{H}^{-1} + \Delta\mathbf{H}^{-1})\mathbf{x} \quad (37)$$

Proof: First note that (36) fulfils the condition in Lemma 4.5 and so $\mathbf{H} - \Delta\mathbf{H}$ is nonsingular with the following inverse expansion

$$\begin{aligned}&(\mathbf{H}-\Delta\mathbf{H})^{-1}\\ &= \mathbf{H}^{-1} + (\mathbf{H}^{-1}\Delta\mathbf{H})\mathbf{H}^{-1} + (\mathbf{H}^{-1}\Delta\mathbf{H})^2\mathbf{H}^{-1} + \ldots\\ &= \mathbf{H}^{-1} + (\mathbf{H}^{-1}\Delta\mathbf{H})\mathbf{H}^{-1} + P((\mathbf{H}^{-1}\Delta\mathbf{H})^2)\mathbf{H}^{-1}\end{aligned} \quad (38)$$

Here the polynomial matrix equation $P(\mathbf{Z})$ is denoted as

$$P(\mathbf{Z}) = \mathbf{Z} + \mathbf{Z}^2 + \mathbf{Z}^3 + \ldots$$

Then multiplying both sides of (38) by $\mathbf{x} \in \Re^n$, we obtain

$$\begin{aligned}&\mathbf{x}^T(\mathbf{H}-\Delta\mathbf{H})^{-1}\mathbf{x}\\ &= \mathbf{x}^T\mathbf{H}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{H}^{-1}\Delta\mathbf{H}\mathbf{H}^{-1}\mathbf{x} + \mathbf{x}^T P((\mathbf{H}^{-1}\Delta\mathbf{H})^2)\mathbf{H}^{-1}\mathbf{x}\\ &\geq \mathbf{x}^T\mathbf{H}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{H}^{-1}\Delta\mathbf{H}\mathbf{H}^{-1}\mathbf{x}\\ &\geq \mathbf{x}^T\mathbf{H}^{-1}\mathbf{x} - \mathbf{x}^T\Delta\mathbf{H}^{-1}\mathbf{x}\end{aligned}$$

by virtue of $\mathbf{H}^{-1}\Delta\mathbf{H}\mathbf{H}^{-1}$ being positive definite. This proves the lower bound. For the upper bound, assume first that the matrix

$$(\mathbf{H}-\Delta\mathbf{H})^{-1} - (\mathbf{H}^{-1} + \Delta\mathbf{H}^{-1}) > 0 \quad (39)$$

i.e. positive definite, then using (38) we have

$$\begin{aligned}
&\left[(\mathbf{H}-\Delta\mathbf{H})^{-1}-(\mathbf{H}^{-1}+\Delta\mathbf{H}^{-1})\right]\Delta\mathbf{H}\\
&=(\mathbf{H}^{-1}+P(\mathbf{H}^{-1}\Delta\mathbf{H})\mathbf{H}^{-1}-(\mathbf{H}^{-1}+\Delta\mathbf{H}^{-1}))\Delta\mathbf{H}\\
&=(P(\mathbf{H}^{-1}\Delta\mathbf{H})\mathbf{H}^{-1}-\Delta\mathbf{H}^{-1})\Delta\mathbf{H}\\
&=P((\mathbf{H}^{-1}\Delta\mathbf{H})^2)-\mathbf{I}>0
\end{aligned}$$

Rearranging and taking norms, we get

$$\begin{aligned}
\|\mathbf{I}\| &< \|P((\mathbf{H}^{-1}\Delta\mathbf{H})^2)\|\\
&< \|\mathbf{H}^{-1}\Delta\mathbf{H}\|^2+\|\mathbf{H}^{-1}\Delta\mathbf{H}\|^3+\ldots\\
&= \frac{\|\mathbf{H}^{-1}\Delta\mathbf{H}\|^2}{1-\|\mathbf{H}^{-1}\Delta\mathbf{H}\|}
\end{aligned} \tag{40}$$

the last line being the sum of an infinite geometric series in $\|\mathbf{H}^{-1}\Delta\mathbf{H}\|<1$. Rearranging (40) and simplifying we get the quadratic form

$$\|\mathbf{H}^{-1}\Delta\mathbf{H}\|^2+\sqrt{n}\|\mathbf{H}^{-1}\Delta\mathbf{H}\|-\sqrt{n}>0 \tag{41}$$

Factorizing and examining the two cases, we find for $n\geq 1$ that (42) holds only for the case

$$\begin{aligned}
\|\mathbf{H}^{-1}\Delta\mathbf{H}\| &> \frac{-n^{\frac{1}{2}}+\sqrt{n+4n^{\frac{1}{2}}}}{2}\\
&> \frac{-n^{\frac{1}{2}}-\sqrt{n+4n^{\frac{1}{2}}}}{2} > \frac{1}{2}
\end{aligned}$$

or

$$\|\Delta\mathbf{H}\| > \frac{1}{2\|\mathbf{H}^{-1}\|}$$

This contradicts condition (36) and so we must have

$$(\mathbf{H}-\Delta\mathbf{H})^{-1}-(\mathbf{H}^{-1}+\Delta\mathbf{H}^{-1})\leq 0 \tag{42}$$

instead of our initial assumption (39). Now take any vector $\mathbf{x}\in\Re^n$ and multiply (42) to give

$$\begin{aligned}
&\mathbf{x}^T\left[(\mathbf{H}-\Delta\mathbf{H})^{-1}-(\mathbf{H}^{-1}+\Delta\mathbf{H}^{-1})\right]\mathbf{x}\leq 0\\
&\Rightarrow \mathbf{x}^T(\mathbf{H}-\Delta\mathbf{H})^{-1}\mathbf{x}\leq \mathbf{x}^T(\mathbf{H}^{-1}+\Delta\mathbf{H}^{-1})\mathbf{x}
\end{aligned}$$

and the upper bound is proved. This completes the proof. ■

The following proposes an upper bound to the maximum possible change in the objective function.

Lemma 4.7 (Sensitivity of Change in the Objective): Suppose for $\boldsymbol{\alpha}'_p \in \mathbb{A}$ the change in objective function is

$$M(\boldsymbol{\alpha}'_p) = \frac{1}{2}\left[{\boldsymbol{\tau}'}_p^T \mathbf{H}_p \boldsymbol{\tau}'_p - \mathbf{v}_p'^T \mathbf{H}_p^{-1} \mathbf{v}'_p\right]$$

Let $\Delta M > 0$ so that the sensitivity for a negative change in the objective function is

$$\begin{aligned} M(\boldsymbol{\alpha}'_p) - \Delta M = \frac{1}{2}\big[&({\boldsymbol{\tau}'}_p - \Delta{\boldsymbol{\tau}'}_p)^T (\mathbf{H}_p - \Delta\mathbf{H}_p)({\boldsymbol{\tau}'}_p - \Delta{\boldsymbol{\tau}'}_p) \\ &-(\mathbf{v}'_p - \Delta\mathbf{v}'_p)^T (\mathbf{H}_p - \Delta\mathbf{H}_p)^{-1}(\mathbf{v}'_p - \Delta\mathbf{v}'_p)\big] \end{aligned} \tag{43}$$

where $\mathbf{H}_p, \Delta\mathbf{H}_p \in \Re^{m\times m}$, $\Delta\mathbf{v}'_p, \Delta{\boldsymbol{\tau}'}_p \in \Re^m$ and

$${\boldsymbol{\tau}'}_p - \Delta{\boldsymbol{\tau}'}_p = (\beta - 1)(\mathbf{H}_p - \Delta\mathbf{H}_p)^{-1}(\mathbf{v}'_p - \Delta\mathbf{v}'_p) \tag{44}$$

Assume further that the perturbation matrix $\Delta\mathbf{H}_p$ is positive definite and

$$\|\Delta\mathbf{H}_p\| < \frac{1}{2\|\mathbf{H}_p^{-1}\|}$$

Then

$$\begin{aligned} \inf -\Delta M \geq -\frac{1}{2}\|\mathbf{H}_p^{-1}\|\|\mathbf{v}'_p - \Delta\mathbf{v}'_p\|^2 \Bigg[&\frac{\|\Delta\mathbf{H}_p\|\|\Delta{\boldsymbol{\tau}'}_p\|^2\rho_B}{\|\Delta\mathbf{H}_p^{-1}\|\|\mathbf{v}'_p - \Delta\mathbf{v}'_p\|^2} \\ &+ \rho_A^2 + \rho_B\Big] \end{aligned} \tag{45}$$

where

$$\rho_A = \frac{\|\Delta\mathbf{v}'_p\|^2}{\|\mathbf{v}'_p - \Delta\mathbf{v}'_p\|^2} \quad \rho_B = \frac{\|\Delta\mathbf{H}_p^{-1}\|}{\|\mathbf{H}_p^{-1}\|}$$

Proof: The condition on $\|\Delta\mathbf{H}_p\|$ allows us to apply (37) of Lemma 4.6 to (43). We obtain

$$\begin{aligned} &M(\boldsymbol{\alpha}'_p) - \Delta M \\ =&\frac{1}{2}\big[({\boldsymbol{\tau}'}_p - \Delta{\boldsymbol{\tau}'}_p)^T (\mathbf{H}_p - \Delta\mathbf{H}_p)({\boldsymbol{\tau}'}_p - \Delta{\boldsymbol{\tau}'}_p) \\ &-(\mathbf{v}'_p - \Delta\mathbf{v}'_p)^T (\mathbf{H}_p - \Delta\mathbf{H}_p)^{-1}(\mathbf{v}'_p - \Delta\mathbf{v}'_p)\big] \\ \geq&\frac{1}{2}\big[({\boldsymbol{\tau}'}_p - \Delta{\boldsymbol{\tau}'}_p)^T (\mathbf{H}_p - \Delta\mathbf{H}_p)({\boldsymbol{\tau}'}_p - \Delta{\boldsymbol{\tau}'}_p) \\ &-(\mathbf{v}'_p - \Delta\mathbf{v}'_p)^T \mathbf{H}_p^{-1}(\mathbf{v}'_p - \Delta\mathbf{v}'_p)\big] \\ &-\frac{1}{2}(\mathbf{v}'_p - \Delta\mathbf{v}'_p)^T \Delta\mathbf{H}_p^{-1}(\mathbf{v}'_p - \Delta\mathbf{v}'_p) \\ \geq&M(\boldsymbol{\alpha}'_p) + {\mathbf{v}'_p}^T \mathbf{H}_p^{-1}\Delta\mathbf{v}'_p - {\boldsymbol{\tau}'}_p^T(\mathbf{H}_p - \Delta\mathbf{H}_p)\Delta{\boldsymbol{\tau}'}_p \\ &-\frac{1}{2}\Big[{\boldsymbol{\tau}'}_p^T \Delta\mathbf{H}_p{\boldsymbol{\tau}'}_p + \Delta{\mathbf{v}'_p}^T \mathbf{H}_p^{-1}\Delta\mathbf{v}'_p \\ &+(\mathbf{v}'_p - \Delta\mathbf{v}'_p)^T \Delta\mathbf{H}_p^{-1}(\mathbf{v}'_p - \Delta\mathbf{v}'_p)\Big] \end{aligned} \tag{46}$$

We now want to show that

$$\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}\Delta\mathbf{v'}_p - \boldsymbol{\tau'}_p{}^T(\mathbf{H}_p - \Delta\mathbf{H}_p)\Delta\boldsymbol{\tau'}_p > 0 \tag{47}$$

Using the definitions in (21) and (44), we get after some algebra

$$\begin{aligned}&\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}\Delta\mathbf{v'}_p - \boldsymbol{\tau'}_p{}^T(\mathbf{H}_p - \Delta\mathbf{H}_p)\Delta\boldsymbol{\tau'}_p\\ &= (2\beta - \beta^2)\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}\Delta\mathbf{v'}_p + (\beta-1)^2\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}\Delta\mathbf{H}_p\mathbf{H}_p^{-1}\mathbf{v'}_p\end{aligned}$$

Now note that we must have

$$\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}(\mathbf{v'}_p - \Delta\mathbf{v'}_p) < 0 \tag{48}$$

to maintain a step towards the minimum of the problem. Using this fact, Lemma 4.4(b) and Lemma 4.3 we then deduce that $(2\beta - \beta^2)\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}\Delta\mathbf{v'}_p > 0$. Then since $\Delta\mathbf{H}_p$ is positive definite, we have shown that (47) holds. This allows us to reduce (46) further to

$$\begin{aligned}-\Delta M \geq -\frac{1}{2}\Big[&\boldsymbol{\tau'}_p{}^T\Delta\mathbf{H}_p\boldsymbol{\tau'}_p + \Delta\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}\Delta\mathbf{v'}_p\\ &+(\mathbf{v'}_p - \Delta\mathbf{v'}_p)^T\Delta\mathbf{H}_p^{-1}(\mathbf{v'}_p - \Delta\mathbf{v'}_p)\Big]\end{aligned} \tag{49}$$

Since all terms in the brackets are positive, we can now deduce that

$$\begin{aligned}\Delta M \leq&\frac{1}{2}\Big[\boldsymbol{\tau'}_p{}^T\Delta\mathbf{H}_p\boldsymbol{\tau'}_p + \Delta\mathbf{v'}_p{}^T\mathbf{H}_p^{-1}\Delta\mathbf{v'}_p\\ &+(\mathbf{v'}_p - \Delta\mathbf{v'}_p)^T\Delta\mathbf{H}_p^{-1}(\mathbf{v'}_p - \Delta\mathbf{v'}_p)\Big]\\ \leq&\frac{1}{2}\|\mathbf{H}_p^{-1}\|\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2\left[\frac{\|\Delta\mathbf{H}_p\|}{\|\mathbf{H}_p^{-1}\|}\frac{\|\Delta\boldsymbol{\tau'}_p\|^2}{\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2}\right.\\ &\left.+\frac{\|\Delta\mathbf{v'}_p\|^2}{\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2} + \frac{\|\Delta\mathbf{H}_p\|}{\|\mathbf{H}_p^{-1}\|}\right]\\ =&\frac{1}{2}\|\mathbf{H}_p^{-1}\|\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2\left[\frac{\|\Delta\mathbf{H}_p\|}{\|\mathbf{H}_p^{-1}\|}\frac{\|\Delta\boldsymbol{\tau'}_p\|^2}{\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2}\right.\\ &\left.+\frac{\|\Delta\mathbf{v'}_p\|^2}{\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2} + \frac{\|\Delta\mathbf{H}_p^{-1}\|}{\|\mathbf{H}_p^{-1}\|}\right]\\ =&\frac{1}{2}\|\mathbf{H}_p^{-1}\|\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2\left[\frac{\|\Delta\mathbf{H}_p\|\|\Delta\boldsymbol{\tau'}_p\|^2\rho_B}{\|\Delta\mathbf{H}_p^{-1}\|\|\mathbf{v'}_p - \Delta\mathbf{v'}_p\|^2}\right.\\ &\left.+\rho_A^2 + \rho_B\right]\end{aligned}$$

Taking the negative, we obtain the infimum for the minimization of the objective function and the lemma is proved. ■

***Remark** 4.2:* Equation (45) relates the sensitivities of the previously mentioned components of the step that affect the rate of convergence namely the sub-gradients, the potential of

overstep and the inverse subHessian. Specifically, the sub-gradient norm gives a quadratic contribution towards the maximum possible change in the objective function. The potential of overstep is related to the sub-gradient through (19) and varies in similar fashion. This explains roughly why current decomposition algorithms e.g. LibSVM, $\mathrm{SVM}^{\mathrm{light}}$ which strictly select working sets based on large sub-gradient norms are an improvement over the original SMO algorithm.

D. Convergence Properties

It may be of some concern that the newer working set selection methods may result in non-convergence of the decomposition algorithms. We show here that as long as the heuristic rule selects working set elements which satisfy a condition, it is sufficient to guarantee asymptotic convergence. Hush and Scovel [22] show that a necessary and sufficient condition for strict decrease in the objective function for SMO type decomposition algorithms is that the working set must consist of violaters and non-bound Support Vectors. We give a generalized convergence theorem for heuristic rules selecting our working sets size and the use of an arbitrary update rule i.e. not necessarily Newton. The condition is sufficient but not necessary.

Lemma 4.8 (Sufficient Conditions): Let the kernel matrix **K**, of (2) be positive definite and denote $\boldsymbol{\alpha'}^*$ as the optimal point of the problem. Then

$$\lim_{t\to\infty} \|\boldsymbol{\alpha'}^t - \boldsymbol{\alpha'}^*\| = 0 \tag{50}$$

if

a) For all $t > 0$ and for all $\boldsymbol{\alpha}'_p \in \mathbb{A}$ we can find $\boldsymbol{\alpha}'_p$ which gives $\Delta\boldsymbol{\alpha}'_p$ such that

$$\Delta\boldsymbol{\alpha'_p}^T \mathbf{v}_p < 0 \tag{51}$$

b) For all $t > 0$ and $\boldsymbol{\alpha}'_p \in \mathbb{A}$ we select

$$\boldsymbol{\alpha}'_p = \{\alpha_i, b | i \in I(\mathbf{E}^+) \cup I(\mathbf{E}^-) \cup I(\boldsymbol{\alpha}_F)\} \tag{52}$$

where $\boldsymbol{\alpha}_F = \{\alpha_i | 0 < \alpha_i < C\}$.

Proof: We will first show that strict decrease in the objective function implies (50) and strict decrease is in turn guaranteed by (51). Then, we will show that the condition (51) requires that we select working sets satisfying (52).

First note that the optimal value of the problem $\Im(\boldsymbol{\alpha'}^*)$ is not at $-\infty$ due to a finite constraint set i.e. **D** is closed,convex and the data set is finite in size. Since **K** is assumed

positive definite, then any subHessian $\mathbf{H}_p$ is at the least positive semidefinite. Since $\Im(\boldsymbol{\alpha})$ is smooth and continuous , it is F-differentiable and the gradient $\mathbf{v}'$, of $\Im(\boldsymbol{\alpha})$ satisfies

$$\begin{aligned}\Im(\boldsymbol{\alpha'}^t) - \Im(\boldsymbol{\alpha'}^*) =& \mathbf{v'}^T(\boldsymbol{\alpha'}^t - \boldsymbol{\alpha'}^*) \\ \leq& C_1\|\boldsymbol{\alpha'}^t - \boldsymbol{\alpha'}^*\| \end{aligned} \tag{53}$$

for some $C_1 > 0$. Now strict decrease in the objective function gives

$$\begin{aligned}&\Im(\boldsymbol{\alpha'}^{t+1}) - \Im(\boldsymbol{\alpha'}^t) < 0 \\ &\Rightarrow \Im(\boldsymbol{\alpha'}^{t+1}) - \Im(\boldsymbol{\alpha'}^*) < \Im(\boldsymbol{\alpha'}^t) - \Im(\boldsymbol{\alpha'}^*)\end{aligned}$$

and at optimality $\Im(\boldsymbol{\alpha'}^t) = \Im(\boldsymbol{\alpha'}^*)$. This in turn implies that as $t \to \infty$ we have

$$\lim_{t\to\infty} \Im(\boldsymbol{\alpha'}^t) - \Im(\boldsymbol{\alpha'}^*) \leq \lim_{t\to\infty} C_1\|\boldsymbol{\alpha'}^t - \boldsymbol{\alpha'}^*\| \to 0$$

Since $C_1 > 0$ then we must have $\lim_{t\to\infty} \|\boldsymbol{\alpha'}^t - \boldsymbol{\alpha'}^*\| = 0$. Now for strict decrease we require

$$\Im(\boldsymbol{\alpha'}^{t+1}) - \Im(\boldsymbol{\alpha'}^t) < 0$$

Using (33), this gives for some working set $\boldsymbol{\alpha}'_p$

$$\begin{aligned}&\Im(\boldsymbol{\alpha'}^{t+1}) - \Im(\boldsymbol{\alpha'}^t) \\ &= \frac{1}{2}\Delta\boldsymbol{\alpha'}^T\mathbf{H}\Delta\boldsymbol{\alpha'} + \Delta\boldsymbol{\alpha'}^T\mathbf{v}' \\ &= \frac{1}{2}\Delta\boldsymbol{\alpha}_{\boldsymbol{p}}'^T\mathbf{H}_p\Delta\boldsymbol{\alpha}'_{\boldsymbol{p}} + \Delta\boldsymbol{\alpha}_{\boldsymbol{p}}'^T\mathbf{v}'_p < 0 \\ &\to \Delta\boldsymbol{\alpha}_{\boldsymbol{p}}'^T\mathbf{v}'_p < -\frac{1}{2}\Delta\boldsymbol{\alpha}_{\boldsymbol{p}}'^T\mathbf{H}_p\Delta\boldsymbol{\alpha}'_{\boldsymbol{p}}\end{aligned}$$

By Lemma 4.3 we then deduce

$$\sup -\frac{1}{2}\Delta\boldsymbol{\alpha}_{\boldsymbol{p}}'^T\mathbf{H}_p\Delta\boldsymbol{\alpha}'_{\boldsymbol{p}} = \sup -\frac{1}{2}\mathbf{v}_p'^T\mathbf{H}_p^{-1}\mathbf{v}'_p = 0$$

and thus conclude

$$\Delta\boldsymbol{\alpha}_{\boldsymbol{p}}'^T\mathbf{v}'_p < 0 \tag{54}$$

is sufficient to guarantee strict decrease in the objective function. This proves (a). To prove (b), note that by Corollary 4.1, we have (51) if the working set elements have indices corresponding to the indices set $I(\mathbf{E}^+) \cup I(\mathbf{E}^-)$. Consider then the set $\boldsymbol{\alpha}_F = \{\alpha_i | 0 < \alpha_i < C\}$. If $v_i = 0$, the element-wise product $\Delta\alpha_i v_i = 0$ and does not affect $\Delta\boldsymbol{\alpha}_{\boldsymbol{p}}'^T\mathbf{v}'_p$. It is possible also to find points with $v_i < 0$ giving $\Delta\alpha_i > 0$ or $v_i > 0$ giving $\Delta\alpha_i < 0$ which result in $\Delta\alpha_i v_i < 0$. Hence, selecting $\boldsymbol{\alpha}'_p$ with indices corresponding to $I(\mathbf{E}^+) \cup I(\mathbf{E}^-) \cup I(\boldsymbol{\alpha}_F)$ ensures (51) holds and is sufficient to guarantee asymptotic convergence. This completes the proof. ■

The conditions in the lemma above are sufficient but not necessary firstly because it is not required to hold for every iteration. Asymptotic convergence can still be achieved if we relax the condition and require it instead to hold for a majority of the iteration. A stronger result can be derived which contains necessary and sufficient conditions for finite termination (as defined e.g. Keerthi et. al [11]), but this will be omitted here since we only require conditions to ensure our algorithm converges. We will make a few remarks. It can be seen that the Newton step satisfies condition Lemma 4.8 (a) provided the step size satisfies Lemma 4.4(a) and we only need to be mindful of condition (b) when designing a heuristic rule. In the event that the kernel matrix is positive semidefinite or indefinite, the Newton step is no longer a guaranteed direction of descent. We can select another working set with a subHessian matrix that is better conditioned i.e. the component corresponding to the kernel matrix is better conditioned. If all possible working sets are exhausted, one may apply a linear direction of descent so that (51) still holds. This last resort may be required in practice due to computational roundoff errors or degenerate data.

V. A Heuristic Framework for the Decomposition Method

In order to solve (18), one can use an algorithm which searches through the entire space of points to find the working set giving the minimum value of (18). We call this the Naive Search Algorithm which tries to satisfy (18) at each iteration. This algorithm if applied to the entire space of possible working sets provides the optimal working set at each iteration.

A. The Naive Search Algorithm

In order to ensure asymptotic convergence as per Lemma 4.8, the search space $\mathbb{A}$ is restricted to the σ-algebra of the set

$$\mathbf{U}= \{\alpha_i | I(\alpha_i) \in I(\mathbf{E}^+) \cup I(\mathbf{E}^-) \cup I(\boldsymbol{\alpha}_F)\}$$

which for practical purposes is ordered as follows

$$\mathbf{U}= \{\alpha_1, \alpha_2, \ldots, \alpha_k | E_1 \geq E_2 \geq \ldots \geq E_k\} \tag{55}$$

The ordered set **U** contains all violaters and non-bound Support Vectors with elements arranged such that the most positive error is at the top and the most negative error is at the bottom. This is practical to implement if we have kept an ordered set of errors, namely $\mathbf{E}^+$ and $\mathbf{E}^-$ (see (32a) and (32b)). A subscript to indicate the iteration is not required since we refer to **U** on a per iteration basis.

ALGORITHM *1 (The Naive Search Algorithm (N1)):*

Initialize $\boldsymbol{\alpha}' = \mathbf{0}$. At each iteration t determine the set $\mathbf{U}$ and let k be the number of elements in $\mathbf{U}$. Set $i, j = 0$.

1. $i = i + 1$ and select $\alpha_1 = \alpha_i \in \mathbf{U}$.
2. $j = j + 1$ and select $\alpha_2 = \alpha_{k-1-j} \in \mathbf{U}$
3. Compute $\Delta\alpha_1, \Delta\alpha_2$.
4. Set $\beta_1 = 1$. If $\alpha_1 + \Delta\alpha_1 < 0$ then $\beta_1 = \frac{-\alpha_1}{\Delta\alpha_1}$ else if $\alpha_1 + \Delta\alpha_1 > C$ then $\beta_1 = \frac{C-\alpha_1}{\Delta\alpha_1}$.
5. Set $\beta_2 = 1$. If $\alpha_2 + \Delta\alpha_2 < 0$ then $\beta_2 = \frac{-\alpha_2}{\Delta\alpha_2}$ else if $\alpha_2 + \Delta\alpha_2 > C$ then $\beta_2 = \frac{C-\alpha_2}{\Delta\alpha_2}$.
6. $\beta = \min\{\beta_1, \beta_2\}$
7. Compute the value of (18). If current value is smaller than previous value, store the indices $\{i, j\}$.
8. Goto Step 2 until $j = k$. If $j = k$ set $j = 0$ and goto Step 1 until $i = k$.
9. Update only α_i, α_j i.e. which minimizes (18). Update *b* by computing Δb using α_i, α_j. Update $\mathbf{U}$. If $\mathbf{U} = \emptyset$ END else $i, j = 0$ and goto Step 1.

The Naive Algorithm searches all possible combinations of working set and updates the one which minimizes (18). The algorithm is clearly going to be slow as the data set size increases due to the increasing combinatorial problem size. This problem would still persist even if a L_1 kernel cache is implemented as in most current algorithms e.g. LibSVM, $\mathrm{SVM}^{\mathrm{light}}$ etc. It becomes worse if a LRU (least recently used) scheme is implemented on a small kernel cache because the rows will be consistently replaced (*cache thrashing*) when searching through the combinations and computing (18).

B. An Adaptive Search Window

One way to reduce the size of the combinatorial problem and also *cache thrashing* is to reduce the dimensionality of the combinatorial problem. A natural reduction has been provided by using (52) of Lemma 4.8 as the search space. A further reduction in dimensionality can be done via a search window method. We first define $\boldsymbol{\Omega}$ as a smaller subset of $\mathbf{U}$ from which the working set elements are chosen. From the sensitivity analysis in Lemma 4.7, our first priority would be to select a working set with a large subgradient norm. This can be done by selecting γ elements off the top and γ elements off the bottom of $\mathbf{U}$. The subset $\boldsymbol{\Omega}$ now becomes a *search window* with size 2γ. We need to also allow for degenerate cases, where sometimes all combinations in $\boldsymbol{\Omega}$ and heuristics fail to produce an acceptable decrease in

the objective function. If this occurs, the next 2γ elements are chosen into $\mathbf{\Omega}$ i.e. the search window is expanded.

The size of $\mathbf{\Omega}$ can be made adaptive by using different values of γ for different stages of the iterations. In the early stages of the iteration, a small γ is sufficient as only an approximately optimal working set is required. As the iteration converges, we would increase γ in order to obtain a more optimal working set. In addition, increasing γ in this manner takes advantage of the operation of the kernel cache. It is well known that in the later stages of the iteration, a subset of points consisting mostly of non-bound Support Vectors are still non-optimal. These points are likely to have their kernel values cached since they have been frequently updated. Therefore a larger search window $\mathbf{\Omega}$ at this stage would also reduce the compromise on computation time since the required values are in the cache. For experimental purposes we propose first a simple two step adaptation defined as below

Heuristic *5.1 (A Simple Adaptive Window):*

$$\begin{aligned} &\text{if} \quad (\mathrm{N_E} > \mathrm{T_G}) \\ &\gamma = L_G \text{ else } \gamma = \mathrm{U_G} \end{aligned}$$

where we use the following settings

$$\begin{aligned} &N_E = \text{Number of Violaters} \\ &T_G = 0.1n \\ &L_G = 5, U_G = 20 \end{aligned} \tag{56}$$

The size of $\mathbf{\Omega}$ now becomes time dependant and this may affect the rate of convergence dramatically in some cases. Better adaptive window techniques are possible and left as further research issues.

C. Shrinking

The shrinking heuristic first introduced by Joachim's [5] and then used in LibSVM excludes updating the gradients or computing the violation of multipliers at the bounds under the assumption that they stay fixed through out the iteration. Optimality of the excluded variables are checked at the end and reoptimization is done if any of them were found to become non-optimal. It is a heuristic meant to improve the run-time speed of the algorithm especially for large data sets. In our framework, we have opted to leave the shrinking heuristic out

for comparison and novelty purposes. This gives us an opportunity to investigate how our proposed heuristics compare against this popular heuristic.

D. Heuristic Framework

In the early stages of the iteration, we believe that it may not be necessary to obtain the working set which optimizes (18) strictly. Instead we can select an approximately optimal working set to compromise between searching for the optimal working set and implementation speed. The introduction of the adaptive search window provides the first step to this approximation. We now investigate several possible heuristics for a second approximation to this.

One way to approximately minimize (18) is to maximize the step magnitude β or to minimize the potential of overstep. This would require that we efficiently predict the direction of motion for each variable. Lemma 4.2 provides us with a rough rule to select a working set that would most likely give us minimum potential of overstep. The following heuristic depicts this.

Heuristic *5.2 (Min Potential of Overstep):*

Select $\alpha_1 \in \mathbf{U}$ from the top and look for $\alpha_2 \in \mathbf{U}$ starting from the bottom such that:

a) if $\alpha_1 = \{0, C\}$ and $E_1 > 0$, select $\alpha_2 = \{0, C\}$ with $E_2 < 0$.

b) if $\alpha_1 = \{0, C\}$ and $E_1 > 0$, select $0 < \alpha_2 < C$ with $E_2 < 0$.

In Heuristic 5.2, the second working set element is selected based on the nature of the first. The heuristic is two-tiered with the first emphasis on updating bounded violaters and the second emphasis on pairing a bounded violater with any other violater. The aim is obtain a pair which has a larger change when updated. We note that a similar heuristic alone has been used in our previous work [20] with some success. Note that the difference in this heuristic compared to [20] is that it is not used for pairs of non-bound Support Vectors. The next heuristic deals with this case and reduces the need to exhaustively compute (18). It approximately selects a better conditioned subHessian.

Heuristic *5.3 (Min η_{12}):*

Assume that in all cases $\eta_{12} > 0$ and at some step t, we get $\mathbf{\Omega} \in I(\boldsymbol{\alpha}_F)$. This automatically excludes the use of Heuristic 5.2.

a) Select $\alpha_1 \in \mathbf{U}$ from the top and look for $\alpha_2 \in \mathbf{U}$ starting from the bottom. Compute η_{12} and find α_1, α_2 which minimizes η_{12}.

b) If using a Gaussian kernel, this can be achieved by maximizing K_{12}.

For the Gaussian kernel, $K_{11} = K_{22} = 1$ for all α_1, α_2 so minimizing η_{12} can be simplified as per Heuristic 5.3 (b).

In some stages of the iteration, the step size may be increased or boosted by selecting $\beta > 1$ e.g. [23]. We refer to this as *acceleration*. The range of β for guaranteed asymptotic convergence in this case has been given by Lemma 4.4 (a). This technique is sometimes known as the Newton relaxation method [24] for the case when $\beta < 1$ and the objective function is a higher order polynomial with several maxima and minima. If we use $\beta < 1$, this reduces the step size and is referred to as *damping*. The optimal use of acceleration and damping in the decomposition setting still remains a research area for us. For now, empirical evidence indicates that acceleration can be applied in the early stages of the iteration for large values of C in order to quickly move bounded variables to opposite bounds. It can also be applied in the stages of the iteration where the step size is relatively too small. We propose the following general heuristic for both α_1 and α_2.

Heuristic *5.4 (Acceleration):*

After applying Heuristic 5.1-5.4, compute $\Delta\alpha_1$ and $\Delta\alpha_2$ without determining β. Then

$$\begin{aligned} &\text{if} \quad (\alpha_{\mathrm{i}} \in \{0, \mathrm{C}\}) \\ &\quad \text{if} \quad (|\Delta\alpha_{\mathrm{i}}| < 0.1\mathrm{C} \quad \text{and} \quad |\Delta\alpha_{\mathrm{i}}| > 0.01\mathrm{C}) \\ &\qquad \beta^* = 1.5 \quad \text{else} \quad \beta^* = 1.3 \end{aligned}$$

Set $\Delta^*\alpha_i = \beta^*\Delta\alpha_i$. Check feasibility of updated iterates and if feasibility violated, scale $\Delta^*\alpha_i$ accordingly.

In the final stages of the iteration, we apply the Naive Algorithm to obtain a more optimal working set. The difference here is that the Naive Algorithm will only be applied to combinations formed from elements in the search window $\mathbf{\Omega}$. Run-time speed should not be compromised too much due to the necessary kernel values being in the kernel cache. Furthermore, computation of (18) can be done with kernel values computed on the fly as to reduce the potential for cache thrashing. The motivation for refining our search in the final stages is to reduce the oscillations between variables that are still difficult to determine. We use the following condition to determine the switch to the Naive Algorithm.

Heuristic *5.5 (Refining the Search via N1):*

$$\text{if} \quad (\text{size}(\mathbf{U}) < 0.1\text{n or } (\max \text{E}_\text{i} - \min \text{E}_\text{j}) < 1.0)$$

Apply N1 to $\mathbf{\Omega}$

E. Termination Criteria

The condition $\max E_i - \min E_j$ described in Heuristic 5.5 is also known as the *violation gap* [25]. It has been used for finite termination of the LibSVM algorithm where the algorithm terminates if the gap size is zero. $\text{SVM}^{\text{light}}$ instead checks that all examples satisfy the KKT conditions within a practical tolerance. One can now see that Heuristic 5.5 operates as the algorithm approaches optimality. Another method is to compare the difference between the primal SVM problem and the dual SVM problem [14], [26]. We have used the first two methods in this framework since it saves us monitoring the primal and dual values and further allows a fair comparison between current algorithms.

F. Block Diagram of the basic Heuristic Framework

We now connect all the previously described heuristics into a single framework. The flowchart depicting how Heuristic 5.1-5.4 are utilitized in the heuristic framework are given in Fig 3. The initialization involves reading the dataset, allocating memory and initializing all necessary variables. The framework is then used for selecting the working set to update at each iteration. The first step is to check for non-optimal points which violate the KKT conditions (8) within a certain practical tolerance ϵ. Then the set $\mathbf{U}$ is determined.

In the "Adaptive $\mathbf{\Omega}$" block, γ is adjusted using Heuristic 5.1 and the corresponding elements selected into the search window $\mathbf{\Omega}$. The "Scan $\mathbf{\Omega}$" block starts by scanning the possible combinations of α_1 and α_2 starting with α_1 corresponding to the most positive E_1 in $\mathbf{\Omega}$ and E_2 the most negative. Heuristic 5.2 is applied if while scanning a pair of violaters at bounds (BSV) i.e. $\alpha_i = \{0, C\}$ are found then Heuristic 5.4 is applied. If all violaters in $\mathbf{\Omega}$ are non-bounded Support Vectors (NBSV) then Heuristic 5.3 is applied. After the pair is selected, the "Update Step" is done. The "Update Step" calculates the new values of the working set, updates all the training errors and also the pseudo Lagrangian, *b*. Heuristic 5.5 is applied during the final stages of the iteration and not shown explicitly in the flow chart. The algorithm then terminates when the termination criteria are met.

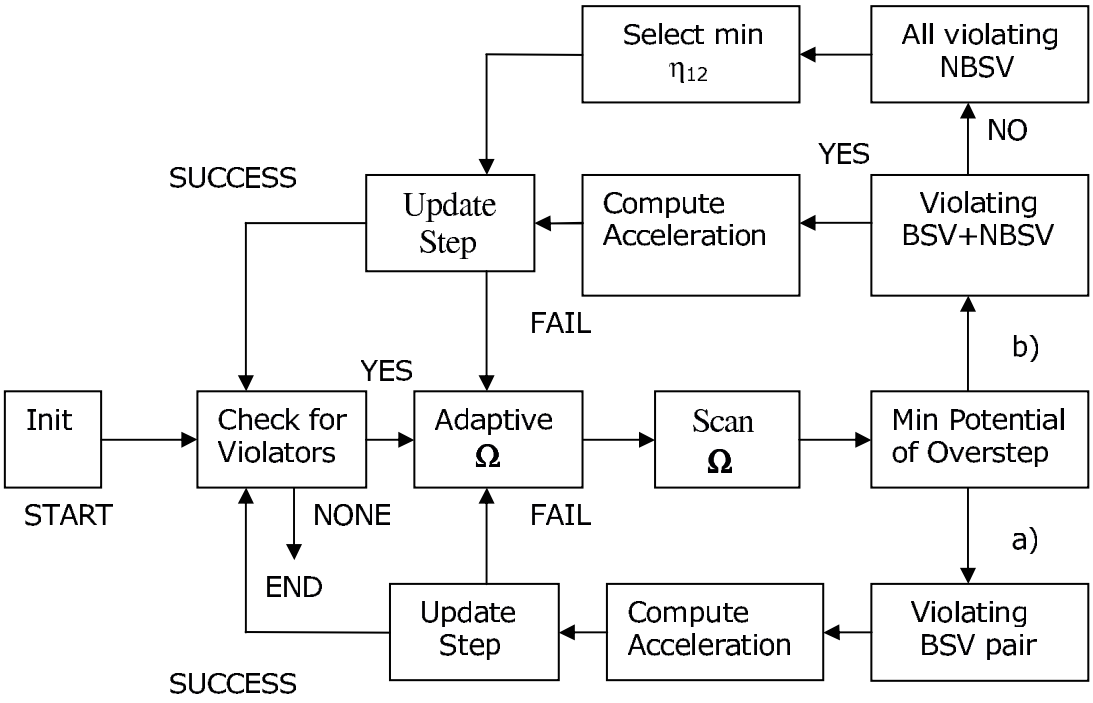

Fig. 3. A flowchart detailing the operation of proposed the heuristic framework

VI. Experimental Method and Discussion

We implement our SVM program called D2C-SVM (Data to Classification) in Visual C++ 6.0 and all experiments were carried out on a Pentium IV, 1.5 GHz computer with 256MB RAM. We compare our implementation against the currently popular optimization software, namely LibSVM (v.2.78) [27] and $\mathrm{SVM}^{\mathrm{light}}$ (v.6.1)[2]. The $\mathrm{SVM}^{\mathrm{light}}$ algorithm uses a working set selection rule which searches for maximal violating pairs and applies an update rule based on steepest search. LibSVM (v.2.78) combines the SMO update rule with a $\mathrm{SVM}^{\mathrm{light}}$ working set selection rule. These two algorithms use the same fixed selection rule throughout the optimization process. Our D2C program implements the heuristic framework in Fig 3 to select working sets. The standard tolerance of $\epsilon = 0.001$ was used for the terminating condition and kernel cache size was set to 120MB for all algorithms.

We record the number of iterations which indicate the length of the sequence of sub-problems successfully solved. The CPU run-times denote the practicality of the algorithm implementations. We now define the following basic ratios which we will use as bench-mark indicators to measure the performance of the D2C heuristic framework against current

[2]http://svmlight.joachims.org/

algorithms. The following definitions will be used

$$T_i = \text{CPU run time in seconds for Algorithm i}$$

$$I_i = \text{Number of iterations for Algorithm i}$$

$$NBSV = \text{Total Free Support Vectors i.e. } 0 < \alpha_j < C$$

$$BSV = \text{Total Support Vectors at bounds i.e. } \alpha_j = C$$

Let CI_{12} be the *iteration ratio* for Algorithm 1 against Algorithm 2 is defined as

$$CI_{12} = \frac{I_1}{I_1 + I_2} \tag{57}$$

The *run-time ratio* CT_{12} for Algorithm 1 against Algorithm 2 is defined as

$$CT_{12} = \frac{T_1}{T_1 + T_2} \tag{58}$$

The performance index μ_{12} for Algorithm 1 against Algorithm 2 is defined as

$$\begin{aligned} \mu_{12} &= \frac{I_1(T_1 + T_2) + T_1(I_1 + I_2)}{2(T_1 + t_2)(I_1 + I_2)} \\ &= \frac{CI_{12} + CT_{12}}{2} \end{aligned} \tag{59}$$

If $\mu_{12} = 0.5$ then we say that both Algorithm 1 and Algorithm 2 have almost similar performance. If $\mu_{12} < 0.5$ we say that Algorithm 1 performs better than Algorithm 2. A value of $\mu_{12} > 0.5$ indicates that Algorithm 2 outperforms Algorithm 1. The performance index μ_{12} can also be interpreted as how well a particular algorithm compares to another in striking a balance between improving the convergence rate (theoretical) and the run-time speed (practical). We also introduce a measure of difficulty of the problem defined as

$$\upsilon = \frac{NBSV}{NBSV + BSV} \tag{60}$$

This is done in the view that the decomposition method has trouble determining the free variables i.e. NBSV [10]. A value closer to unity indicates a problem that is more difficult. In all cases, Algorithm 1 is taken as our D2C heuristic framework and comparisons made against the other state of the art decomposition algorithms. We present some initial performance results over some popular benchmark data sets.

A. Performance over UCI benchmark data

We first test our algorithm on the UCI benchmark data sets which have been widely used to benchmark decomposition algorithms e.g. SMO [4]. The benchmark dataset we use is the

UCI 1 to 9 adult dataset [28] with sizes approximately ranging from 1000-40000 points. For illustration purposes, we compare the D2C algorithm against the full Naive Algorithm (N1) on the first three Adult datasets (see Table II). Then four sets of benchmarks were done comparing our algorithm against $\mathrm{SVM}^{\mathrm{light}}$ and LibSVM. The performance results are reported in Table III-VI.

B. Run Time performance over kernel and C parameters

In this experiment, we employ the large UCI adult 9 dataset and use the Gaussian kernel. For the Gaussian kernel, we vary the kernel width σ^2 over a range $1 \geq \sigma^2 \geq 100$ across a range of SVM parameters of C where $0.1 \leq C \leq 100$. The results are plotted on a log-scale graph in Fig 4-Fig 5 for comparison against LibSVM and $\mathrm{SVM}^{\mathrm{light}}$.

C. Performance over poorly conditioned data

The purpose of this experiment is to show that choosing a working set with a well conditioned subHessian results in improved theoretical convergence rates. We employ the UCI Web data set for this experiment which consists of 8 datasets in increasing size. These datasets contain examples with no attributes as well as missing attributes resulting in an overall poorly condition Hessian matrix. The performance results are reported in Table VII-X.

D. Discussion of Results

From Table II, it is clear that the Naive Search Algorithm chooses the optimal working set for (18) as depicted by the lower number of iterations. However, as expected the computational run-time increases tenfold, until it becomes impractical to search for the optimal working set at each iteration. In the benchmarks on the UCI Adult datasets, the heuristics in our algorithm achieve on average a 30% reduction in the number of iterations compared to LibSVM and $\mathrm{SVM}^{\mathrm{light}}$ (see Table III-VI). The proposed Heuristics 5.2-5.5 provided a better approximation to the optimal working set rule instead of only searching for maximal violating pairs. For the poorly conditioned data sets, our D2C algorithm converges in approximately one third of the number of iterations than LibSVM and $\mathrm{SVM}^{\mathrm{light}}$. This indicates that improving the approximation to the optimal value of (18) at each iteration increases the theoretical convergence rate (see Table VII-X).

For the run-time performance, we observed that our heuristic framework was on average faster than the shrinking heuristic used by LIBSVM and $\mathrm{SVM}^{\mathrm{light}}$ for the benchmarks on the

UCI adult data set (Table III-VI). The results in Fig 4-Fig 5 seem to confirm this further when we test the largest dataset over a range of Gaussian kernel and C parameters sometimes achieving an improvement over an order of magnitude. This is surprising considering the fact that the shrinking heuristic causes both LIBSVM and $\mathrm{SVM}^{\mathrm{light}}$ to update a smaller set of gradients as convergence is reached. The overall effect should be a shorter training time for larger datasets. We note that the two benchmarks on the UCI Adult set test both extremes of the problem difficulty ν_{12}, so the number of free variables is not a contributing factor to this observation as noted by [10]. In contrast however, the shrinking heuristic resulted in faster run time for LIBSVM and $\mathrm{SVM}^{\mathrm{light}}$ over the UCI web data set benchmarks (Table VII-X). Even though, the D2C algorithm selected a more optimal working set, its slower run-time performance resulted in a poorer performance index, μ_{12} against the other algorithms. Again, it can be seen that the problem difficulty ν_{12} did not seem to be a contributing factor.

On further investigation, we found that the shrinking heuristic was useful for problems where only a small subset of multipliers were updated during the entire iteration. In the UCI Adult benchmarks, we noted that on average 70% of the variables i.e. Lagrangian multipliers, were updated and the kernel cache was fully utilized. This is in stark contrast to the UCI Web benchmarks where on average only 25% of the points were updated (the other 75% do not violate the constraints during optimization). Since variables that are ”shrunk” need their gradients to be recomputed and changes to be stored, a larger percentage of changing points and inaccurate shrinking would naturally require more computational time and slow the algorithm down. We deduce then that in these situations, the shrinking heuristic could be replaced with Heuristic 5.2-5.5 to obtain a better run-time performance. In retrospect, if only a small fraction of points are updated during the optimization then Heuristic 5.2-5.5 should be coupled with the shrinking heuristic to improve overall algorithm performance.

Heuristic 5.2 and 5.3 try to obtain the working set which approximately solves (18) within the search window controlled by Heuristic 5.1. The acceleration applied in Heuristic 5.4 boosts small step sizes but we observed that in Table VI and Table X where there is a significant range in problem difficulty ν_{12}, acceleration worked better for smaller values of ν_{12}. This observation can be explained as follows. A more difficult problem has a large number of working sets composed of free Support Vectors. As convergence is approached, it is less likely that the free Support Vectors will become bounded Support Vectors. The feasible regions of the subproblems corresponding to these working sets then look more and

more like $\Re^n$ and hence an unconstrained Newton step i.e. $\beta = 1$ provides the optimal step size. Applying acceleration in these cases will result in a suboptimal step and hence slow convergence. We summarize the merits and demerits of the individual heuristics discussed in Table I.

We note that several other optimization techniques modify the original cost function by introducing barrier and penalty functions [29] into the cost to account for constraint violation. These methods have been minimally explored in the decomposition setting but they already face problems if the optimal solution contained variables at bounds as these methods prevent the variables from reaching the boundaries. To this end, we point out that the diversity of datasets makes using any one fixed heuristic rather myopic in foresight. The extra degree of freedom in choosing the working set at each iteration affords an algorithm which should adapt to the problem. We believe that this is increasingly important when applying the Support Vector Machine to online applications. In this work, we have described a heuristic framework which has the initial workings of an adaptive algorithm. Further research will concentrate on expanding the adaptive nature of this framework in order to increase the effectiveness of the decomposition method for training Support Vector Machines.

VII. Conclusion

In this paper, we describe a heuristic framework for the decomposition method for training Support Vector Machines. We first provide some theoretical analysis concerning the working set problem. Based on these results, we then proposed a series of heuristics which not only

TABLE I

Comparison of advantages and disadvantages of heuristics

Heuristic	Advantage	Disadvantage
Shrinking	Improves implementation run-time if majority of points stay optimal.	Slow if recomputation of gradient occurs often. May require more iterations if wrong points are shrunk.
Adaptive Search Window	Reduces search space for working set.	Success dependant on adaptation of window size.
Min Potential Overstep	Assists in obtaining working sets with larger step sizes.	Does not work if all violaters are NBSV.
Min η_{12}	Approximates the Naive Algorithm.	Does not work well if data is degenerate.
Acceleration/Damping	Boosts small step sizes in initial and final iterations.	Does not work well if in final iterations only NBSVs remain.
Naive Algorithm	Requires fewer number of iterations.	Very slow as problem size increases.

select a better sequence of working sets but which also improves algorithm run times. The use of heuristics is further motivated by the fact that improving the theoretical convergence rates resulted in a degradation of run-time performance. Benchmark tests reveal that the proposed heuristics can increase the performance of the decomposition algorithm in terms of iterations and computation run-time compared to current algorithms.

REFERENCES

[1] V. N. Vapnik, *The nature of statistical learning theory*, 2nd ed., ser. Statistics for engineering and information science. New York: Springer, 2000.

[2] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, "Incremental training of Support Vector Machines," *to appear in IEEE Transactions on Neural Networks*, 2005.

[3] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: An application to face detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 130–136.

[4] J. Platt, "Fast training of Support Vector Machines using sequential minimal optimization," in *Advances in Kernel Methods-Support Vector Learning*, B. Schlkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge MIT Press, 1998, pp. 185–208.

[5] T. Joachims, "Making large scale support vector machine learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schlkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge , MIT Press, 1998, pp. 169–184.

[6] C. Hsu and C. Lin, "A simple decomposition method for Support Vector Machines." *Machine Learning*, vol. 46, pp. 291–314. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/bsvm/

[7] C. C. Chang, C. W. Hsu, and C. J. Lin, "The analysis of decomposition methods for Support Vector Machines," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, pp. 1003–1008, 2000.

[8] C. J. Lin, "On the convergence of the decomposition method for Support Vector Machines," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1288–1298, 2001.

[9] ——, "Linear convergence for a decomposition method for Support Vector Machines," November 2001.

[10] P. Laskov, "Feasible direction decomposition algorithms for training Support Vector Machines," *Machine Learning*, vol. 46, no. 1-3, pp. 315–349, 2002. [Online]. Available: citeseer.ist.psu.edu/laskov01feasible.html

[11] S. S. Keerthi and E. G. Gilbert, "Convergence of a generalized SMO algorithm for SVM classifier design," *Machine Learning*, vol. 46, no. 1-3, pp. 351–360, 2002. [Online]. Available: citeseer.nj.nec.com/keerthi00convergence.html

[12] D. Lai, N. Mani, and M. Palaniswami, "Effect of constraints on sub-problem selection for solving Support Vector Machines using space decomposition," in *The 6th International Conference on Optimization: Techniques and Applications (ICOTA6) accepted*, Ballarat,Australia, 2004.

[13] G. H. Golub and C. F. V. Loan, *Matrix Computations 2nd Ed*. London: The John Hopkins University Press, 1990.

[14] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines : and other kernel- based learning methods*. New York: Cambridge University Press, 2000.

[15] R. Fletcher, *Practical methods of optimization*. Chichester [Eng.] ; New York: J. Wiley, 1981.

[16] R. E. Bellman, *Dynamic programming*. Princeton,: Princeton University Press, 1957.

[17] E. Polak, *Optimization: Algorithms and Consistent Approximations*. Springer Verlag: Applied Mathematical Sciences, 1997.

[18] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*. London ; New York: Academic Press, 1981.

[19] K. Chong and H. Stanislaw, *An introduction to Optimization*. New York: Wiley-Interscience Series in Discrete Mathematics and Optimization, 2001.

[20] D. Lai, N. Mani, and M. Palaniswami, "A new momentum minimization decomposition method for Support Vector Machines," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, 2004, pp. 2001–2006.

[21] A. Ben-Israel and T. N. Greville, *Generalized Inverses: Theory and Applications 2nd Ed*, ser. Canadian Mathematical Society. New York: Springer, 2003.

[22] D. Hush and C. Scovel, "Polynomial-time decomposition algorithms for support vector machines," 2000. [Online]. Available: citeseer.ist.psu.edu/hush00polynomialtime.html

[23] D. Lai, M. Palaniswami, and N. Mani, "An extrapolated sequential minimial optimization algorithm for Support Vector Machines," in *International Conference on Intelligent Sensing and Information Processing*. Chennai,India: 415-420, 2004.

[24] edited by Heinz-Otto Peitgen., *Newtons method and dynamical systems*. Dordrech [Netherlands];Boston: Kluwer Academic, 1989.

[25] C. J. Lin, "A formal analysis of stopping criteria of decomposition methods for Support Vector Machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1045–1052, 2002.

[26] B. Schlkopf and A. J. Smola, *Learning with kernels : Support Vector Machines, regularization, optimization, and beyond*, ser. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 2002.

[27] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for Support Vector Machines*, 2001, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[28] C. Blake and C. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[29] D. Du and J. Sun, *Advances in optimization and approximation*, ser. Nonconvex optimization and its applications ; vol. 1. Kluwer Academic, 1994.

TABLE II

COMPARISON OF PERFORMANCE FOR TRAINING UCI ADULT BENCHMARKS ON GAUSSIAN KERNEL WITH $C = 1$ AND $\sigma^2 = 10$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS THE NAIVE SEARCH ALGORITHM (N1)

DataSet	Size	T_1	T_2	I_1	I_2
Adult 1	1605	0.48	56.53	611	598
Adult 2	2265	1.00	166.65	965	871
Adult 3	3185	1.92	609.81	1271	1210

TABLE III

COMPARISON OF PERFORMANCE FOR TRAINING UCI ADULT BENCHMARKS ON GAUSSIAN KERNEL WITH $C = 1$ AND $\sigma^2 = 10$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS LIBSVM

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
1605	0.48	0.63	611	956	0.39	0.44	0.41	0.15
2265	1.00	1.52	965	1552	0.38	0.40	0.39	0.16
3185	1.92	3.34	1271	1943	0.40	0.36	0.38	0.14
4781	4.33	9.45	1835	2848	0.39	0.31	0.35	0.12
6414	7.89	19.80	2423	4078	0.37	0.29	0.33	0.12
11220	25.39	56.31	4364	7442	0.37	0.31	0.34	0.10
16100	53.91	94.17	6434	10283	0.38	0.36	0.37	0.09
22697	103.42	169.25	9563	15709	0.38	0.38	0.38	0.09
32561	217.22	329.48	14934	24738	0.38	0.40	0.39	0.08

TABLE IV

COMPARISON OF PERFORMANCE FOR TRAINING UCI ADULT BENCHMARKS ON LINEAR KERNEL WITH $C = 0.05$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS LIBSVM

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
1605	0.41	0.39	627	979	0.39	0.51	0.45	0.94
2265	0.81	1.08	860	1625	0.35	0.43	0.39	0.95
3185	1.66	2.59	1247	2114	0.37	0.39	0.38	0.96
4781	3.72	7.52	1722	2620	0.40	0.33	0.36	0.97
6414	6.50	26.88	2187	3419	0.39	0.19	0.29	0.97
11220	20.80	47.59	3794	5456	0.41	0.30	0.36	0.98
16100	42.19	75.56	5289	6522	0.45	0.36	0.40	0.99
22697	86.25	146.47	7037	11558	0.38	0.37	0.37	0.99
32561	162.98	253.63	9397	15700	0.37	0.39	0.38	0.99

TABLE V

COMPARISON OF PERFORMANCE FOR TRAINING UCI ADULT BENCHMARKS ON GAUSSIAN KERNEL WITH $C = 1$ AND $\sigma^2 = 10$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS SVM$^{\text{light}}$

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
1605	0.48	0.86	611	822	0.43	0.36	0.39	0.15
2265	1.00	1.99	965	1487	0.39	0.33	0.36	0.16
3185	1.92	3.58	1271	1765	0.42	0.35	0.38	0.14
4781	4.33	7.85	1835	2575	0.42	0.36	0.39	0.12
6414	7.89	15.97	2423	3886	0.38	0.33	0.36	0.12
11220	25.39	97.97	4364	6893	0.39	0.21	0.30	0.10
16100	53.91	175.64	6434	9714	0.40	0.23	0.32	0.09
22697	103.42	333.58	9563	14504	0.40	0.24	0.32	0.09
32561	217.22	672.58	14934	21331	0.41	0.24	0.33	0.08

TABLE VI

COMPARISON OF PERFORMANCE FOR TRAINING UCI ADULT BENCHMARKS ON LINEAR KERNEL WITH $C = 0.05$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS SVM$^{\text{light}}$

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
1605	0.41	0.17	627	819	0.43	0.70	0.57	0.94
2265	0.81	0.52	860	1200	0.42	0.61	0.51	0.95
3185	1.66	1.05	1247	1858	0.40	0.61	0.51	0.96
4781	3.72	2.4	1722	2483	0.41	0.61	0.51	0.97
6414	6.50	4.18	2187	3071	0.42	0.61	0.51	0.97
11220	20.80	13.21	3794	4635	0.45	0.61	0.53	0.98
16100	42.19	30.08	5289	7015	0.43	0.58	0.51	0.99
22697	86.25	60.06	7037	10281	0.41	0.59	0.50	0.99
32561	162.98	129.39	9397	14196	0.40	0.56	0.48	0.99

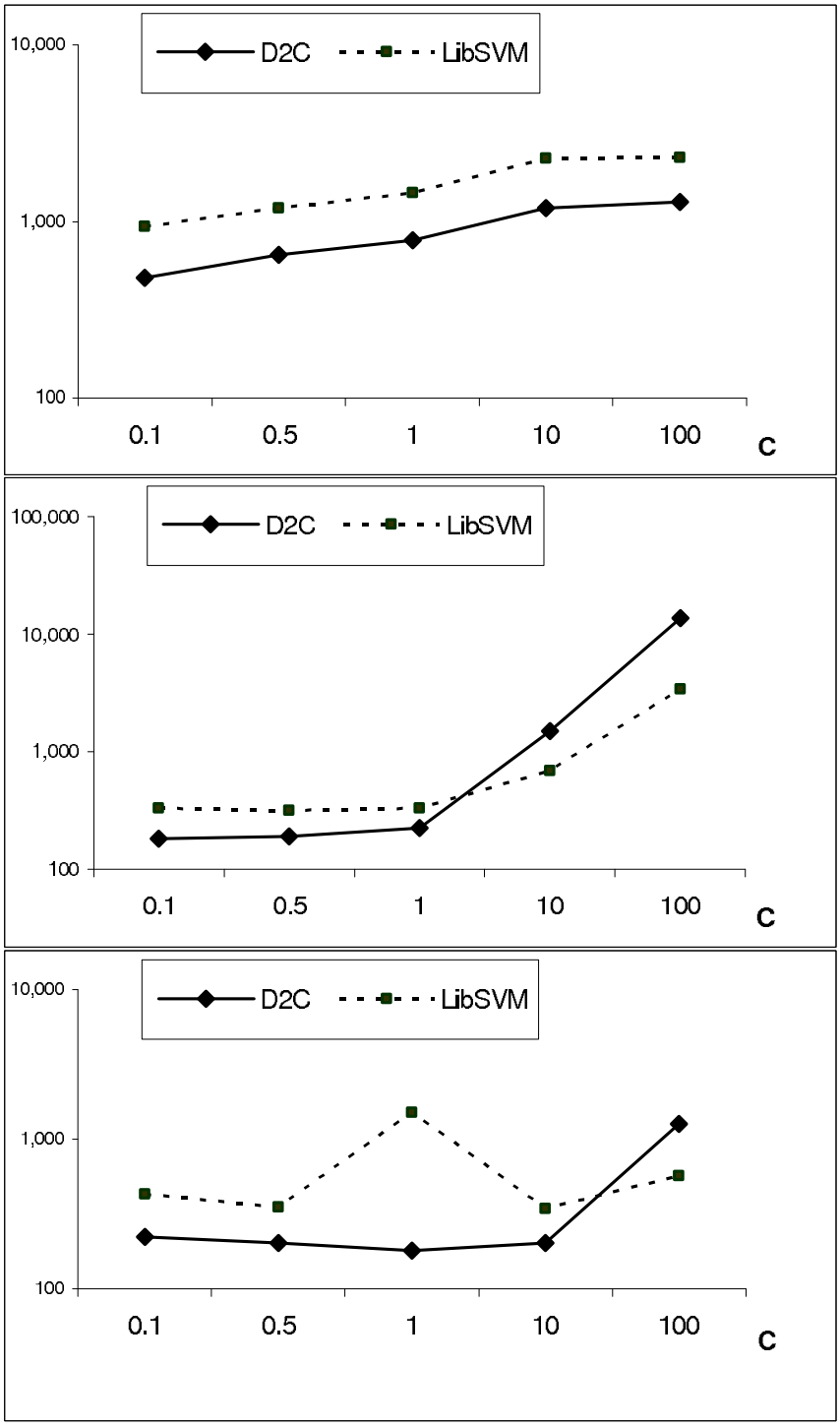

Fig. 4. Comparison of run-time performance (seconds) against LibSVM for training UCI Adult 9 using Gaussian kernel over a range of C. Left to Right: $\sigma^2 = 1, 10, 100$.

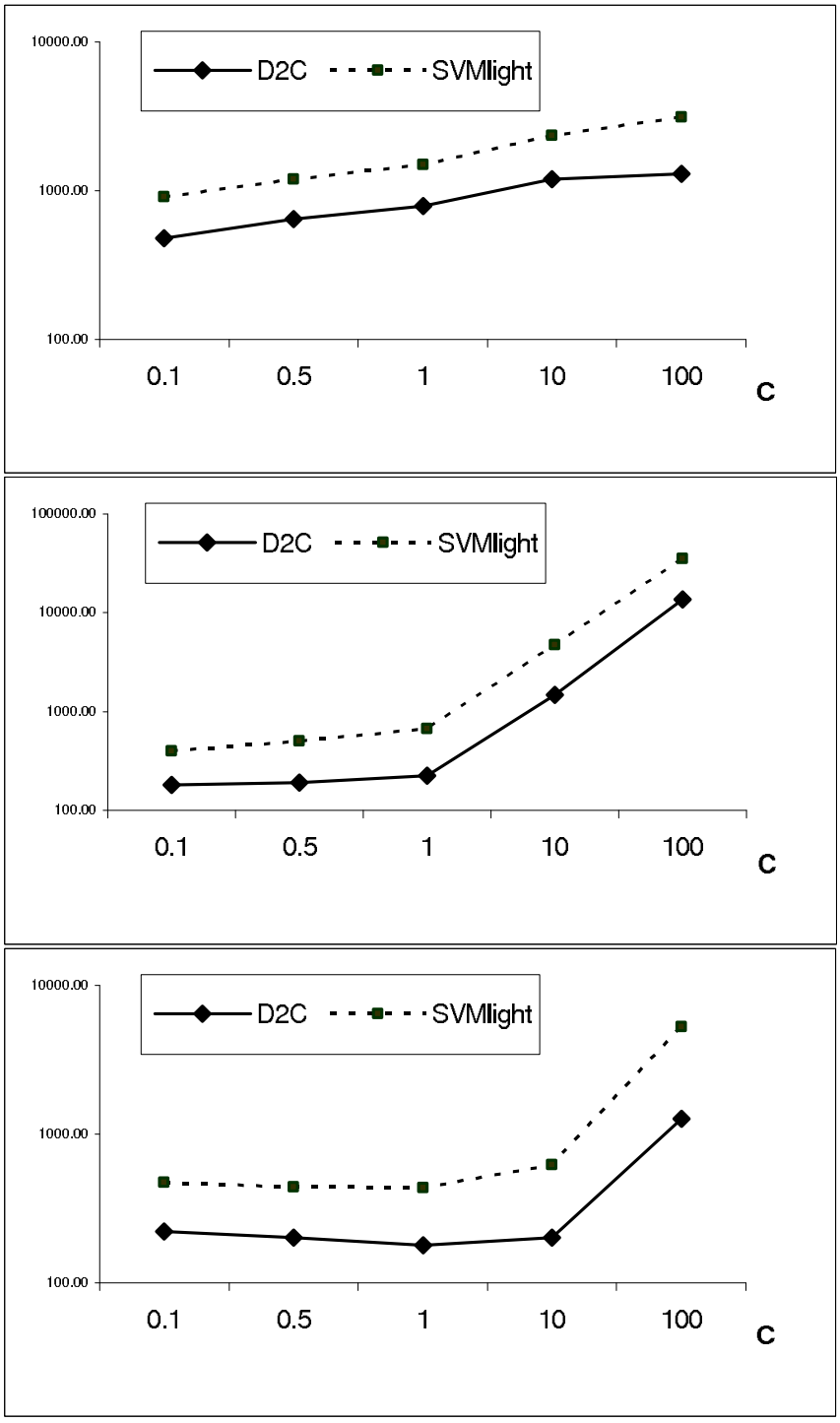

Fig. 5. Comparison of run-time performance (seconds) against $\mathrm{SVM}^{\mathrm{light}}$ for training UCI Adult 9 using Gaussian kernel over a range of C. Left to Right: $\sigma^2 = 1, 10, 100$.

TABLE VII

COMPARISON OF PERFORMANCE FOR TRAINING UCI WEB BENCHMARKS ON GAUSSIAN KERNEL WITH $C = 5$ AND $\sigma^2 = 10$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS LIBSVM

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
2477	2.13	0.63	1738	3104	0.36	0.77	0.57	0.89
3470	4.20	1.20	2396	3481	0.41	0.78	0.59	0.88
4912	6.19	2.17	2833	3803	0.43	0.74	0.58	0.86
7366	11.83	4.80	4143	6402	0.39	0.71	0.55	0.85
9888	18.61	8.39	5260	7390	0.42	0.69	0.55	0.84
17188	44.92	57.31	8913	11206	0.44	0.44	0.44	0.78
24692	91.20	107.39	11886	13519	0.47	0.46	0.46	0.76
49749	580.77	373.50	18263	20840	0.47	0.61	0.54	0.69

TABLE VIII

COMPARISON OF PERFORMANCE FOR TRAINING UCI WEB BENCHMARKS ON LINEAR KERNEL WITH $C = 1$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS LIBSVM

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
2477	0.84	0.27	1811	5161	0.26	0.76	0.51	0.75
3470	2.19	0.63	3217	8698	0.27	0.78	0.52	0.66
4912	6.64	1.09	6646	12475	0.35	0.86	0.60	0.60
7366	12.55	2.59	9863	37744	0.21	0.83	0.52	0.51
9888	25.28	4.69	13435	32956	0.29	0.84	0.57	0.43
17188	41.97	15.55	20269	103225	0.16	0.73	0.45	0.31
24692	87.73	32.11	30503	146846	0.17	0.73	0.45	0.24
49749	337.84	187.72	48340	235591	0.17	0.64	0.41	0.15

TABLE IX

COMPARISON OF PERFORMANCE FOR TRAINING UCI WEB BENCHMARKS ON GAUSSIAN KERNEL WITH $C = 5$ AND $\sigma^2 = 10$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS $\text{SVM}^{\text{light}}$

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
2477	2.13	1.44	1738	2764	0.39	0.60	0.49	0.89
3470	4.20	2.56	2396	3187	0.43	0.62	0.53	0.88
4912	6.19	4.3	2833	3803	0.43	0.59	0.51	0.86
7366	11.83	10.63	4143	6402	0.39	0.53	0.46	0.85
9888	18.61	22.75	5260	7390	0.42	0.45	0.43	0.84
17188	44.92	73.63	8913	11206	0.44	0.38	0.41	0.78
24692	91.20	143.86	11886	13519	0.47	0.39	0.43	0.76
49749	580.77	551.58	18263	20640	0.47	0.51	0.49	0.69

TABLE X

COMPARISON OF PERFORMANCE FOR TRAINING UCI WEB BENCHMARKS ON LINEAR KERNEL WITH $C = 1$. ALGORITHM 1 IS D2C AND ALGORITHM 2 IS $\text{SVM}^{\text{light}}$

Size	T_1	T_2	I_1	I_2	CI_{12}	CT_{12}	μ_{12}	ν_{12}
2477	0.84	0.52	1811	5552	0.25	0.62	0.43	0.75
3470	2.19	0.85	3217	7857	0.29	0.72	0.51	0.66
4912	6.64	1.47	6646	11591	0.36	0.82	0.59	0.60
7366	12.55	3.85	9863	29194	0.25	0.77	0.51	0.51
9888	25.28	4.41	13435	21219	0.39	0.85	0.62	0.43
17188	41.97	17.92	20269	63469	0.24	0.70	0.47	0.31
24692	87.73	34.39	30503	67856	0.31	0.72	0.51	0.24
49749	337.84	106.83	48340	107681	0.31	0.76	0.53	0.15