Department of Electrical
and
Computer Systems Engineering

Technical Report
MECSE-29-2003

Increasing the step of the Newtonian Decomposition Method for Support Vector Machines

D.Lai, N.Mani and M.Palaniswami

MONASH
UNIVERSITY

Increasing the step of the Newtonian Decomposition Method for Support Vector Machines

*D.Lai , *N.Mani, +M.Palaniswami
*Dept. of Electrical and Computer Systems Engineering
Monash University, Clayton, Vic. 3168, Australia.
+Dept. of Electrical and Electronic Engineering,
The University of Melbourne, Vic. 3010, Australia.
{daniel.lai,n.mani@eng.monash.edu.au, swami@ee.mu.oz}

Abstract: The Newtonian method is a standard iterative method with quadratic convergence rates for solving large optimization problems. The Support Vector Machine problem formulation requires the solution of large datasets, which is ideal for the application of Newtonian methods. We point out that the Sequential Minimal Optimization (SMO) algorithm is a Newtonian Decomposition Method that is popular due to its efficiency but unfortunately does not scale too well with large data sets. The algorithm is an implementation of the decomposition method, which solves a sequence of sub problems instead of the entire problem at once. In this report, we introduce an extrapolation parameter to the SMO update method and investigate the effect on the rate of convergence of this algorithm. We first show that the SMO update method is Newtonian and that extrapolation ensures the update is norm reducing on the objective function. We also investigate the bounds of extrapolation and derive optimal estimates for this parameter. It was observed that choosing the working set pair according to some partial order does result in slightly faster speedups in algorithm performance.

I. INTRODUCTION

The Support Vector Machines (SVM) developed by Vapnik[1] and co-workers has been shown to be a powerful supervised learning tool. The standard soft-margin Support Vector Machine is a binary classifier applied to classify a data set defined as,

$$\begin{aligned} &\Theta=\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)...(\mathbf{x}_l, y_l)\} \\ &\mathbf{x}_i \in \mathbb{R}^n \\ &y_i = \{1,-1\} \end{aligned} \tag{1}$$

The SVM formulation is essentially a regularized minimization problem leading to the use of Lagrange 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 of the hyperplane. This idea is further extended for data that is not linearly separable; where it is first mapped via a nonlinear function to a higher dimension feature space. Maximizing the margin of the hyperplane in either space is equivalent to maximizing the distance between the class boundaries. Vapnik[1] suggests that the form of the hyperplane, $f(\mathrm{x}) \in \mathrm{F}$ be chosen from family of functions with sufficient capacity. In particular, F contains functions for the linearly and non-linearly separable hyperplanes;

$$f(x) = \sum_{i=1}^{l} w_i x_i + b \tag{2}$$

$$f(x) = \sum_{i=1}^{l} w_i \phi(x_i) + b \tag{3}$$

The weight vector, **w** in (3) is no longer the same expansion as in the linearly separable case (2). In fact, the non-linear mapping $\phi : \mathbf{x} \subset \Re^n \to \Re^m$ and $n, m \in [1, \infty)$ defines the mapping from data space to feature space. Hence the weights in feature space will have a one to one correspondence with the elements of $\phi(\mathbf{x})$. Now for separation in feature space, we would like to obtain the hyperplane with the following properties;

$$
\begin{aligned}
&f(\mathbf{x}) = \sum_{i=1}^{l} w_i \varphi(\mathbf{x}_i) + b \\
&f(\mathbf{x}) > 0 \quad \forall \quad i : y_i = +1 \\
&f(\mathbf{x}) < 0 \quad \forall \quad i : y_i = -1
\end{aligned} \tag{4}
$$

The conditions in (4) can be described by a strict linear discriminant function, so that for each element pair in Θ, we require that;

$$
y_i \left(\sum_{i=1}^{l} w_i \varphi(\mathbf{x}_i) + b \right) \geq 1 \tag{5}
$$

The distance from the hyperplane to a support vector is $\frac{1}{\|\mathbf{w}\|}$ and the distance between the support vectors of one class to the other class is simply $\frac{2}{\|\mathbf{w}\|}$ by geometry. The soft-margin minimization problem relaxes the strict discriminant in (5) by introducing slack variables, ξ_i and is formulated as;

$$
\text{P.1} \begin{cases} \text{minimize} \quad \Im(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{l} w_i^2 + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \begin{cases} y_i \left(\sum_{i=1}^{l} w_i \varphi(\mathbf{x}_i) + b \right) \geq 1 + \xi_i \\ \forall i = 1..l \end{cases} \end{cases} \tag{6}
$$

We now apply Lagrange Theory to solve (6) giving us the Lagrange Primal problem of the following form;

$$
\text{P.2} \begin{cases} \text{minimize} \quad \Im(\mathrm{w}, \alpha) = \frac{1}{2} \sum_{i=1}^{l} w_i^2 + \sum_{i=1}^{l} \alpha_i \left(y_i \left(\sum_{j=1}^{l} w_j \varphi(x_j) + b \right) - 1 - \xi_i \right) + \sum_{i=1}^{l} \pi_i \xi_i \\ \text{subject to} \begin{cases} \nabla \left(\frac{1}{2} \sum_{i=1}^{l} w_i^2 \right) + \nabla \left(\sum_{i=1}^{l} \alpha_i \left(y_i \left(\sum_{j=1}^{l} w_j \varphi(x_j) + b \right) - 1 - \xi_i \right) \right) = 0 \\ \alpha_i, \pi_i \geq 0 \quad \forall \mathrm{i}=1..l \end{cases} \end{cases} \tag{7}
$$

The following dual optimization of a cost function written in terms of Lagrange Multipliers alone is usually implemented by incorporating the gradient condition into the cost function of P.2 and minimizing the following Lagrange dual in terms of Lagrange multipliers, α alone;

$$
\text{P.3} \begin{cases} \min \quad \underset{\alpha \in \mathbb{Q} \subset \Re^l}{\Im(\boldsymbol{\alpha})} = \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{l} \alpha_i \\ \text{where} \quad \mathbb{Q} = \left\{ \alpha \,\middle|\, 0 \leq \alpha_i \leq C, \sum_{i=1}^{l} \alpha_i y_i = 0, \forall i = 1..l \right\} \end{cases} \tag{8}
$$

The separating hyperplane surface in (4) can now be written in terms of Lagrange Multipliers;

$$
f(x) = sign \left(\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b \right) \tag{9}
$$

In this report, we point out that the SMO is in fact a Newton-method solving a different cost function than the problem originally stated in [2]. It was postulated by Chang [3] that update rules using the steepest gradient

direction might not give the optimal rate of convergence for a decomposition method. This leads us to investigate applying a larger Newtonian step controlled by an extrapolation parameter, which we will introduce in the following. We investigate applying our extrapolation parameter to the SMO update rule to increase the size of the step towards the minimum of the cost function. We also introduce the notion of *damping* and *extrapolation* regions to the SMO iteration by allowing our extrapolation parameter to vary and observe the resulting algorithm performance. We note here that several modern implementations of SMO and SVMlight such as [4] and LibSVM[5] are currently much faster than the original pseudocodes. However, the newer implementations run a basic combination of the original update rules using several advance heuristics and more efficient coding. Our work here investigates an improvement to the basic SMO update rule. We could apply this technique to modern implementations and investigate their convergences rates further.

The outline of this paper is as follows; in the next section, we review the Sequential Minimal Optimization algorithm and rewrite it in the familiar Newton update form. In Section 3, we try to justify the use of extrapolation to increase the speed of a general Newton update rule and further investigate its use with the bounded Newton form of SMO. The remaining sections will be left for our results and further discussions.

II. Sequential Minimal Optimization (SMO): A Newtonian Decomposition Method

We review the update rule used in SMO[2] which we show to be a hybrid element-wise Newton method in a decomposition algorithm setting. For the sake of simplicity, we will adopt the notations used by Platt in the original implementation for this summary. The working set size in SMO is fixed to two in order to enforce the equality constraint in (8) at every step of the iteration. We denote the working set as;

$$\alpha_W = \{\alpha_1, \alpha_2\} \quad \alpha_1, \alpha_2 \in \Re \tag{10}$$

Whenever we update the value of two multipliers, SMO maintains the equality constraint by ensuring;

$$\begin{aligned} &\alpha_1 y_1 + \alpha_2 y_2 = k \\ &\text{where} \quad \begin{cases} \alpha_1 + \alpha_2 = k & \text{if } y_1 = y_2 \\ \alpha_1 - \alpha_2 = k & \text{if } y_1 \neq y_2 \end{cases} \end{aligned} \tag{11}$$

We make use of (11) to express the following for some iterative step, t where $t > 0$;

$$\begin{aligned} s &= y_1 y_2 \\ \alpha_1^{t+1} + s\alpha_2^{t+1} &= \alpha_1^{\,t} + s\alpha_2^{\,t} = \gamma \end{aligned} \tag{12}$$

Platt defines the error of a training point as E_i where,

$$E_i = f(x_i) - y_i \quad \forall i = 1..l \tag{13}$$

The minimum of the cost function (8) subject to the equality constraint is first found in the direction of an arbitrary α_2 while the other multiplier α_1 is set to a value that maintains the equality constraint. We can derive the update rule for SMO by writing the cost function in terms of the working set members;

$$\Im(\alpha_1, \alpha_2) = -\alpha_1 - \alpha_2 + \frac{1}{2}K_{11}\alpha_1^{\,2} + \frac{1}{2}K_{22}\alpha_2^{\,2} + sK_{12}\alpha_1\alpha_2 + y_1\alpha_1 v_1 + y_2\alpha_2 v_2 + \Im_{const} \tag{14}$$

$$\text{where} \quad v_i = \sum_{j=3}^{l} y_j \alpha_j^{\,t} K_{ij} = f(x_1) - y_1\alpha_1^{\,t}K_{1i} - y_1\alpha_2^{\,t}K_{2i}$$

$$\Im_{const} = -\sum_{i=3}^{l}\alpha_i + \frac{1}{2}\sum_{i,j=3}^{l}\alpha_i\alpha_j y_i y_j K(x_i, x_j)$$

Then, the objective function expressed in α_2 entirely is,

$$\Im(\alpha_2) = -\gamma + s\alpha_2 - \alpha_2 + \frac{1}{2}K_{11}(\gamma - s\alpha_2)^2 + \frac{1}{2}K_{22}\alpha_2{}^2 + sK_{12}(\gamma - s\alpha_2)\alpha_2 + y_1(\gamma - s\alpha_2)v_1 + y_2\alpha_2 v_2 + \Im_{const} \tag{15}$$

The minimum of (15) can be found with respect to α_2, by taking the derivatives;

$$\nabla\Im(\alpha_2) = sK_{11}(\gamma - s\alpha_2) - K_{22}\alpha_2 + K_{12}\alpha_2 - sK_{12}(\gamma - s\alpha_2) - y_2v_2 + y_2v_1 - s + 1 = 0 \tag{16}$$

$$\nabla(\nabla\Im(\alpha_2)) = 2K_{12} - K_{22} - K_{11} \tag{17}$$

Expanding (16) and substituting for s and γ, we get

$$\alpha_2^{t+1}(K_{11} + K_{22} - 2K_{12}) = \alpha_2^t(K_{11} + K_{22} - 2K_{12}) + y_2(f(x_1) - f(x_2) + y_2 - y_1) \tag{18}$$

Finally, after some rearranging we obtain the update rule,

$$\alpha_2^{t+1} = \alpha_2^t + y_2\frac{(E_1 - E_2)}{(K_{11} + K_{22} - 2K_{12})} \tag{19}$$

In order to ensure that $\alpha_2 \in \mathbb{Q}$, we enforce the following;

$$\alpha_2^{t+1} = \begin{cases} UB & \text{if } \alpha_2^{t+1} \geq UB \\ \alpha_2^{t+1} & \text{if } LB < \alpha_2^{t+1} < UB \\ LB & \alpha_2^{t+1} \leq LB \end{cases} \tag{20}$$

where;

$$UB = \begin{cases} \min(C, \alpha_1 + \alpha_2) & \text{if } y_1 = y_2 \\ \min(C, C + \alpha_2 - \alpha_1) & \text{if } y_1 \neq y_2 \end{cases}$$
$$LB = \begin{cases} \max(0, \alpha_1 + \alpha_2 - C) & \text{if } y_1 = y_2 \\ \max(0, \alpha_2 - \alpha_1) & \text{if } y_1 \neq y_2 \end{cases} \tag{21}$$

The update for α_1 is then found by using (12);

$$\alpha_1^{t+1} = \alpha_1^t + s(\alpha_2^t - \alpha_2^{t+1,bounded}) \tag{22}$$

It can be seen here that the equality constraint is enforced at each step of the iteration through the use of (22). We now show that the update rule is Newtonian by first expanding (19);

$$\alpha_2^{t+1} = \alpha_2^t + y_2\frac{(E_1 - E_2)}{(K_{11} + K_{22} - 2K_{12})} = \alpha_2^t + \frac{(y_2f(x_1) - y_2y_1 - y_2f(x_2) + 1)}{(K_{11} + K_{22} - 2K_{12})} \tag{23}$$

Now, we simplify (16),

$$\begin{aligned}
&\nabla\Im(\alpha_2) \\
&= sK_{11}(\gamma - s\alpha_2) - K_{22}\alpha_2 + K_{12}\alpha_2 - sK_{12}(\gamma - s\alpha_2) - y_2v_2 + y_2v_1 - s + 1 \\
&= sK_{11}(\alpha_1 + s\alpha_2 - s\alpha_2) - K_{22}\alpha_2 + K_{12}\alpha_2 - sK_{12}(\alpha_1 + s\alpha_2 - s\alpha_2) - y_2\left(f(x_2) + b - y_1\alpha_1K_{12} - y_2\alpha_2K_{22}\right) \\
&\quad + y_2\left(f(x_1) + b - y_1\alpha_1K_{11} - y_1\alpha_2K_{21}\right) - s + 1 \\
&= s\alpha_1K_{11} - K_{22}\alpha_2 + K_{12}\alpha_2 - s\alpha_1K_{12} - y_2f(x_2) + s\alpha_1K_{12} + \alpha_2K_{22} + y_2f(x_1) - s\alpha_1K_{11} + \alpha_2K_{21} - s + 1 \\
&= y_2f(x_1) - y_2f(x_2) - s + 1 \\
&= y_2f(x_1) - y_1y_2 - y_2f(x_2) + 1
\end{aligned}$$

Then using (17) we get the required relationship;

$$\begin{aligned}\alpha_2^{t+1} &= \alpha_2^{t} + \frac{(y_2 f(x_1) - y_2 y_1 - y_2 f(x_2) + 1)}{(K_{11} + K_{22} - 2K_{12})} \\ &\rightarrow \alpha_2^{t} - \frac{\nabla \Im(\alpha_2)}{\nabla(\nabla \Im(\alpha_2))}\end{aligned} \tag{24}$$

Since α_2 is arbitrary, we get the following relationship for α_1;

$$\begin{aligned}\nabla \Im(\alpha_2) &= y_2 (E_1 - E_2) \\ &= -y_1 y_1 y_2 (E_2 - E_1) \\ &= -s y_1 (E_2 - E_1) \\ &= -s \nabla \Im(\alpha_1)\end{aligned} \tag{25}$$

We note that (17) implies $\nabla(\nabla \Im(\alpha_2)) = \nabla(\nabla \Im(\alpha_1))$ since α_2 is arbitrary. Then, we have from (22) the following result;

$$\begin{aligned}\alpha_1^{t+1} &= \alpha_1^{t} + s(\alpha_2^{t} - \alpha_2^{t+1,bounded}) \\ &= \alpha_1^{t} + s(\alpha_2^{t} - \alpha_2^{t} + \frac{\nabla \Im(\alpha_2)}{\nabla(\nabla \Im(\alpha_2))}) \\ &= \alpha_1^{t} + s\frac{\nabla \Im(\alpha_2)}{\nabla(\nabla \Im(\alpha_2))} \\ &= \alpha_1^{t} - s^2 \frac{\nabla \Im(\alpha_1)}{\nabla(\nabla \Im(\alpha_1))} = \alpha_1^{t} - \frac{\nabla \Im(\alpha_1)}{\nabla(\nabla \Im(\alpha_1))} \rightarrow \alpha_1^{t+1} \in \mathbb{Q}\end{aligned} \tag{26}$$

Then the update rule on the working set members can be written as;

$$\alpha_2^{t+1} = \alpha_2^{t} - \frac{\nabla \Im(\alpha_2)}{\nabla(\nabla \Im(\alpha_2))} \tag{27}$$

$$\alpha_1^{t+1} = \alpha_1^{t} - \frac{\nabla \Im(\alpha_1)}{\nabla(\nabla \Im(\alpha_1))} \tag{28}$$

We note from (8) that $\alpha_1, \alpha_2 \in \mathbb{Q}$ implies that the multipliers are bounded at each iterative stage to ensure that they remain in the feasible region. Therefore, depending on the choice of $\mathbb{Q}$, the updates are optimal in the direction of the two multipliers on the feasible region. The standard Newton form[6] with the objective function $F(\alpha)$ is given by;

$$\alpha^{t+1} = \alpha^{t} - \frac{F(\alpha^{t})}{\nabla F(\alpha^{t})} \tag{29}$$

By comparison, the forms of (27) and (28) seem to imply that from a Newtonian perspective, the objective function is instead the derivative of (8) by setting $F(\alpha) = \nabla \Im(\alpha)$. This is now similar to the problem solved by Joachim's and others (see [5, 7]) where the derivative of the Lagrange Dual is solved to make computations easier and to avoid computing the quadratic function.

III. EXTRAPOLATION: INCREASING THE STEP TOWARDS THE MINIMUM

A. Extrapolation on the general Newton Method

We now introduce an extrapolation parameter ζ to the general Newton iteration form below;

$$\alpha_i^{t+1} = \alpha_i^{t} - \zeta \frac{F(\alpha_i)}{\nabla F(\alpha_i)} \qquad \forall i>0 \tag{30}$$

We note here that when $\zeta = 1$, we recover the normal Newton step as in (29). We will show that the introduction of the extrapolation parameter to the general Newton method still ensures the iteration rule is norm reducing on the cost function. We then show that this extrapolation parameter is bounded to an interval when applied to a general Newton method. We first make use of a well-known result that we adapt in the following technical lemma;

Lemma III.1 (Ortega[8])

Let the function $F : \mathbb{Q} \subset \Re^N \to \Re^M$ be Gateux-differentiable (G-differentiable) on the convex set $\mathbb{Q} \in \Re^N$. Then for any vector $x, y \in \mathbb{Q}$, we have the following;

$$\|Fy - Fx\| \le \sup_{0\le t\le 1} \left\|\nabla F\left(x + t(y-x)\right)\right\| \|y - x\| \tag{31}$$

Proof:

First, assume that the supremum is bounded from above, i.e.

$$S = \sup_{0\le t\le 1} \left\|\nabla F\left(x + t(y-x)\right)\right\| < \infty$$

Then for an arbitrary $\varepsilon > 0$, we can choose a $t \in [0,1]$ such that

$$\begin{aligned}
\left\|F\left(x + t(y-x)\right) - Fx - t\nabla F\left(x + t(y-x)\right)(y-x)\right\| &\le \varepsilon t \|y-x\| \\
\left\|F\left(x + t(y-x)\right) - Fx\right\| &\le \sup_{0\le t\le 1} t\left\|\nabla F\left(x + t(y-x)\right)(y-x)\right\| + \varepsilon t\|y-x\| \\
&\le St\|y-x\| + \varepsilon t\|y-x\|
\end{aligned}$$

Since F is G-differentiable on a convex set, it is also continuous in t so that for any γ the following holds,

$$\left\|F\left(x + \gamma(y-x)\right) - Fx\right\| \le S\gamma\|y-x\| + \varepsilon\gamma\|y-x\| \tag{32}$$

When $\gamma = 1$, we recover (31)

$$\begin{aligned}
\|Fy - Fx\| &\le S\|y-x\| + \varepsilon\|y-x\| \\
&\le \sup_{0\le t\le 1}\left\|\nabla F\left(x + t(y-x)\right)\right\|\|y-x\| + \varepsilon\|y-x\| \\
&\le \sup_{0\le t\le 1}\left\|\nabla F\left(x + t(y-x)\right)\right\|\|y-x\|
\end{aligned}$$

Assume $\gamma < 1$ and since ∇F exists everywhere on the convex set, there is a λ such that $\lambda \in (\gamma, 1)$ and we have;

$$\begin{aligned}
&\left\|F\left(x + \lambda(y-x)\right) - F\left(x + \gamma(y-x)\right) - \nabla F\left(x + \lambda(y-x)\right)(\lambda-\gamma)(y-x)\right\| \le \varepsilon(\lambda-\gamma)\|y-x\| \\
&\to \left\|F\left(x + \lambda(y-x)\right) - F\left(x + \gamma(y-x)\right)\right\| \le (\lambda-\gamma)\left\|\nabla F\left(x + \lambda(y-x)\right)\right\|\|(y-x)\| + \varepsilon(\lambda-\gamma)\|y-x\| \\
&\qquad\qquad \le \left(\left\|\nabla F\left(x + \lambda(y-x)\right)\right\| + \varepsilon\right)(\lambda-\gamma)\|y-x\|
\end{aligned}$$

But from (32), we have;

$$\left\|F\left(x+\lambda\left(y-x\right)\right)-Fx\right\| \le \left(\left\|\nabla F\left(x+\lambda\left(y-x\right)\right)\right\|+\varepsilon\right)\gamma\left\|y-x\right\|$$

This implies that $\lambda > \gamma > 1$ which contradicts our previous assumption of λ, γ and so (31) must hold □

The proof of Lemma III.1 makes use of the definition of G-differentiability of functions on a convex set, which is defined in the Appendix. We now show that extrapolation ensures that the update rule is norm reducing in the following theorem.

Theorem III.I

Let the function $F : \mathbb{Q} \subset \Re^N \to \Re^N$ be G-differentiable at a point x and $\nabla F\left(x\right)$ be invertible. Then there exists a scalar $\zeta > 0$ such that for any vector $x \in \mathbb{Q}$ and y=$\dfrac{\mathrm{F(x)}}{\nabla F\left(x\right)}$;

$$\left\|F\left(x-\zeta y\right)\right\| \le \left\|Fx\right\| \tag{33}$$

Proof:

Since F is G-differentiable, we have the following relationship;

$$F\left(x-t\zeta y\right)-Fx+\psi t\zeta y \le \nabla Fx \tag{34}$$

Here $\psi \in L\left(\Re^n, \Re^m\right)$ is some linear operator and we get equality only at $t = 0$. We now define an arbitrary norm on $\Re^N$ and rearranging (34) obtain the relationship;

$$\left\|F\left(x-t\zeta y\right)-Fx-\nabla Fx\right\| \le -t\zeta\left\|\psi y\right\|$$

Let $t = 1$ and we now have for some $\varepsilon > 0$;

$$\left\|F\left(x-\zeta y\right)-Fx\right\| \le \left\|F\left(x-\zeta y\right)\right\|-\left\|Fx\right\| \le -\varepsilon\zeta\left\|\psi y\right\|$$
$$\to \left\|F\left(x-\zeta y\right)\right\| \le \left\|Fx\right\|-\varepsilon\zeta\left\|\psi y\right\|$$

Since ε is an arbitrary positive value, clearly (33) holds if and only if $\zeta > 0$. □

However, extrapolation is only norm-reducing provided that the parameter is positive which gives us a lower bound on the range of possible values. We now compute the upper bound for this parameter, by showing in the following theorem that the norm reduction property holds only if ζ lies in a bounded interval.

Theorem III.II

Let the function $F : \mathbb{Q} \subset \Re^N \to \Re^N$ be G-differentiable at the points $x, y \in \mathbb{Q}$ and suppose the following holds,

$$\left\|\nabla F\left(x-ty\right)-\nabla F\left(x\right)\right\| \le \psi t\left\|y\right\| \qquad \forall \mathrm{t} \in \left[0,1\right] \tag{35}$$

then (33) holds for all $\zeta \in \left[0, \dfrac{\left\|Fx\right\|}{\psi\left\|y\right\|^2}\right]$.

Proof:

Since F is G-differentiable, then (31) in Lemma III.1 gives the following for any $t \in \left[0,1\right]$;

$$\left\|F\left(x-\zeta y\right)-Fx\right\| \le \sup_{0\le t\le 1} t\left\|\nabla F\left(x-t\left(x-\zeta y\right)\right)\right\|\left\|x-\zeta y\right\| \tag{36}$$

We have;

$$\begin{aligned}\sup_{0\le t\le 1} t\left\|\nabla F\left(x-t\left(x-\zeta y\right)\right)\right\|\left\|x-\zeta y\right\| &= \sup_{0\le t\le 1} t\left\|\nabla F\left(x-t\left(x-\zeta y\right)\right)\left(x-\zeta y\right)\right\| \\ &= \sup_{0\le t\le 1} t\left\|x\nabla F\left(x-t\left(x-\zeta y\right)\right)-\zeta y\nabla F\left(x-t\left(x-\zeta y\right)\right)\right\| \\ &\le \sup_{0\le t\le 1} t\left\|\zeta y\nabla F\left(x-t\left(x-\zeta y\right)\right)\right\| \\ &\le \sup_{0\le t\le 1} t\zeta\left\|y\right\|\left\|\nabla F\left(x-\zeta ty\right)-\nabla F\left(tx\right)\right\|\end{aligned}$$

Then from (36) and noting that $\zeta > 0$, we have

$$\begin{aligned}\left\|F\left(x-\zeta y\right)-Fx\right\| &\le \sup_{0\le t\le 1} t\zeta\left\|y\right\|\left\|\nabla F\left(x-\zeta ty\right)-\nabla F\left(tx\right)\right\| \\ &\le \sup_{0\le t\le 1} t\zeta\left\|y\right\|\left\|\nabla F\left(x-ty\right)-\nabla F\left(tx\right)\right\|\end{aligned} \tag{37}$$

Let $t=1$, then if (33) holds, we have the following upper bound;

$$\begin{aligned}\left\|F\left(x-\zeta y\right)-Fx\right\| &\le \zeta\left\|y\right\|\left\|\nabla F\left(x-ty\right)-\nabla F\left(x\right)\right\| \\ &\le \zeta\psi\left\|y\right\|^2 \le \left\|Fx\right\| \\ \rightarrow \zeta &\le \frac{\left\|Fx\right\|}{\psi\left\|y\right\|^2}\end{aligned}$$

Thus for Theorem III.I to hold ζ is constrained to the interval $\left[0,\frac{\left\|Fx\right\|}{\psi\left\|y\right\|^2}\right]$. We define the linear operator ψ to be identity matrix I when using the Euclidean norm, so that the element wise update rule is simplified. The bounds for the extrapolation parameter ζ can now simplified to $0<\zeta<\frac{\nabla^2 Fx}{Fx}$ □

The norm reducing theorems we derived above are in terms of minimization of the cost function and the domain space where the update rule is a mapping on the entire domain. It is trivial to show that they hold verbatim for maximizations of the cost function by simply replacing $F\left(x\right)$ with its negative and rewriting them to show that extrapolation is also norm maximizing. The theorems above show that extrapolation can increase the step to the minimum of the cost function for a Newtonian method by ensuring the update is norm reducing. In the following, we will test our proposed extrapolation on the standard SMO algorithm and measure the improvement in performance over some benchmark datasets. The extrapolated SMO update rule for the updates of the working set is now just;

$$\alpha_w^{t+1}=\begin{pmatrix}\alpha_1^{t+1}\\ \alpha_2^{t+1}\end{pmatrix}=\begin{pmatrix}\alpha_1^{t}\\ \alpha_2^{t}\end{pmatrix}-\begin{pmatrix}\frac{\zeta y_1(E_2-E_1)}{\eta}\\ \frac{\zeta y_2(E_1-E_2)}{\eta}\end{pmatrix} \tag{38}$$

IV. Experimental Method

In the series of experiments, a Pentium IV, 1.5 GHz computer with 256MB RAM was used. Our proposed idea was implemented in the form of an extrapolated SMO on Visual C++ 6.0 with kernel caching. We recorded the number of iterations and epochs required till convergence was established. Here iterations refer to the number of successful updates on the Lagrange multipliers and epochs refer to the number of loops which the algorithm goes

through in the main program. We have kept our algorithm consistent with the original SMO in order to test the effect of our proposed extrapolation on the update rule. Thus, we need only to compare the number of iterations and epochs to determine the effectiveness of our extrapolated SMO. The CPU times are different from Platt's report due to the different architecture, which this experiment is executed on. We ran our proposed SMO on the UCI adult data set [9] and fish data set with a tolerance of 0.001. All results have been averaged out over 10 runs in order to minimize the effects of random ordering imposed by SMO heuristics on the speed of the algorithm.

For the UCI datasets we set C=1 and applied a Gaussian kernel for separation. We investigated the algorithm performance over the range $0.8 < \zeta < 1.8$ and report the results for the first 3 datasets in Fig 1. We also observed the behavior of the objective function (8) comparing the effect of ζ on the number of epochs required till algorithm termination. We then ran a series of experiments of UCI adult set 1-5 at a higher tolerance to investigate the effect of extrapolation on the sensitivity of the algorithm to violating points. Next, we report the results for training a SVM to classify the UCI web 1 and web 2 (Fig 5) data using a Gaussian kernel with C=5.

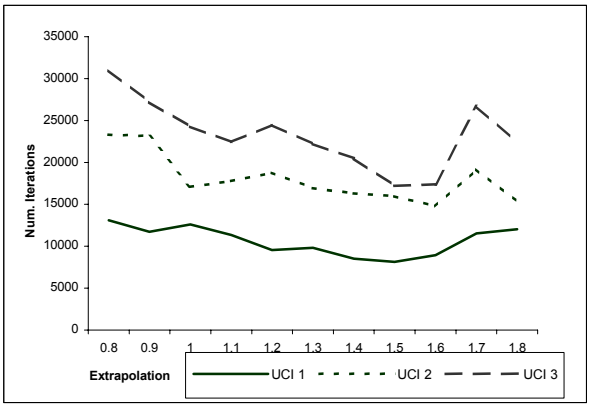

Fig 1: Number of iterations required for SMO across 3 datasets. Damped(ζ<1) and extrapolation(ζ>1) regions shown here.

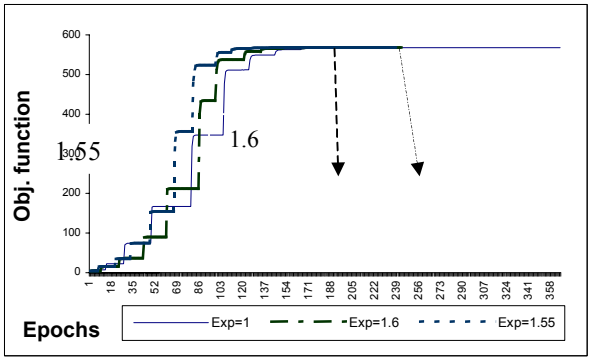

Fig 2: Behaviour of the Wolfe objective function for classical SMO ($\zeta = 1$) and the extrapolated SMO. Arrows indicate the points of termination for the algorithm and the respective extrapolation parameters.

Extrapolation	UCI 1	UCI 2	UCI 3	UCI 4	UCI 5
0.8	13112	22453	30980	52851	76868
0.9	11726	22557	27208	46656	62693
1	12621	21455	24258	40409	72384
1.1	11340	17657	22439	37320	60704
1.2	9567	19372	24455	38859	57713
1.3	9818	15100	22183	35841	47314
1.4	8514	17973	20462	35256	55683
1.5	8151	15700	17209	32405	54063
1.6	8944	17070	17400	33391	53454
1.7	11537	18073	26704	36302	47912
1.8	12037	15884	22370	38567	53838

Table 1: Number of iterations till convergence using extrapolation for UCI adult datasets 1-5 at a tolerance of 10^{-4}

We then examined the effect of arranging or sorting the order of support vectors in the cache to be chosen for update. This implicitly imposes some form of partial ordering on the working set. We sorted the Support Vectors in order of increasing and decreasing magnitude and compare the resulting algorithm speed against the random choice of maximal violators. The results are shown for number of iterations in Fig 3 while Fig 4 shows the best improvement in speed when sorting the Support Vectors in decreasing order. We then tested the speed of the algorithm across a range of C for an arbitrary UCI dataset; in this case we show the results for UCI 3 in Fig 6.

We also ran our fish dataset across a range of kernels since it was highly inseparable. We have found that C=1 produced an acceptable accuracy on the test set [10] when we tested this proposed algorithm using the Gaussian and Linear Kernels. For the Gaussian kernel, when σ=10, we discovered that ζ lay in the range of $0.5 < \zeta < 2.0$ and $\zeta > 2.0$ caused the iteration to oscillate and not converge. Furthermore, we obtained a large improvement of about 50% less iterations required using extrapolated SMO against the normal SMO for this kernel Fig 7. It was noted by Lin[11] that SMO may be slow for linear kernels from the results of his co-workers. We thus applied a linear kernel to the data and observed in Fig 8 an improvement of about 25% in the speed of the algorithm.

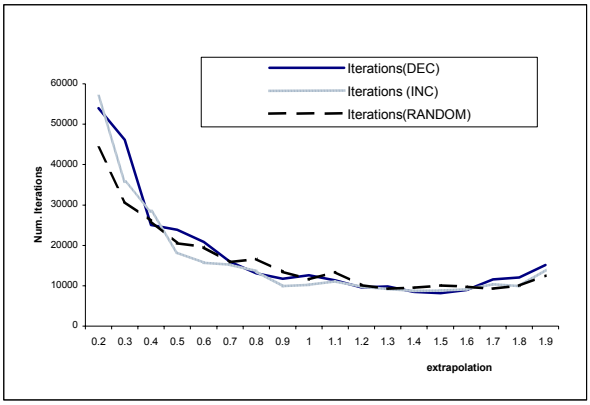

Fig 3: Number of sucessful updates for extrapolated and damped SMO using random, increasing, decreasing cache sorting on UCI 1.

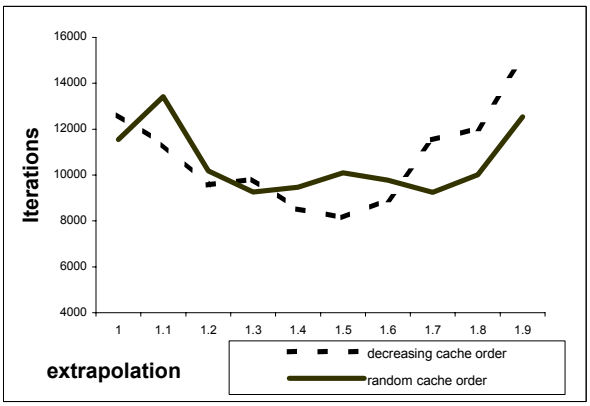

Fig 4: Performance of SMO in extrapolation on UCI 1 for two cache modes.

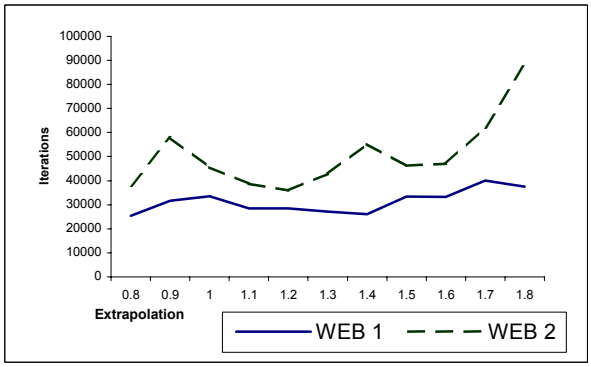

Fig 5: Extrapolated SMO on UCI web1 (2477 examples) and web2(3470 examples)

V. DISCUSSION

Platt inadvertently proposed a decomposition algorithm with a Newton iterate rule and solved the problem of applying a Newton rule to large datasets without the memory constraint problem. However, we believe that the rates of convergence of SMO are not quadratic like Newton methods due to the decomposition aspect of the problem. In fact, a worse case convergence rate has been shown to be linear[12]. We have shown that the update rule of SMO is simply just a bounded form of Newtonian iteration with clever heuristics besides being related to Bergman methods as claimed by [2]. From Fig 3, we can observe that choosing values of $\zeta < 1$ results in a *damped* SMO where the update rule is not optimal in a Newtonian sense. Empirical results show that values of $\zeta < 1$ (larger damping constant) not too far away from unity do give faster rates of convergence. However, if ζ is too small, the rate of convergence becomes very much slower and is no longer practical. We say the algorithm is over-damped and convergence is very slow. We found that when testing on UCI datasets, at $\zeta = 0.1$ the algorithm took excessively long to converge and for smaller values it had not converged after 5 hours.

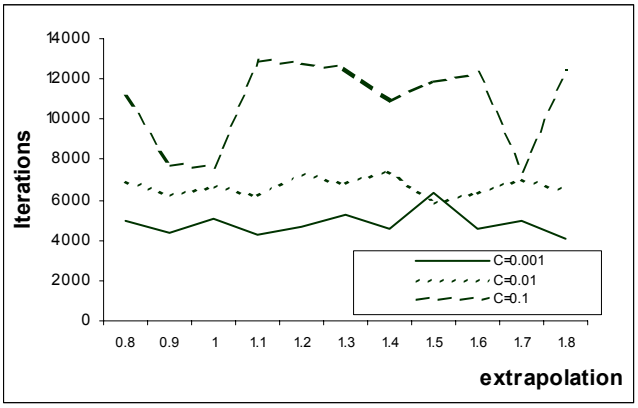

Fig 6: Extrapolated-SMO against normal SMO(ζ=1) for UCI 3 data set over a range of C parameters.

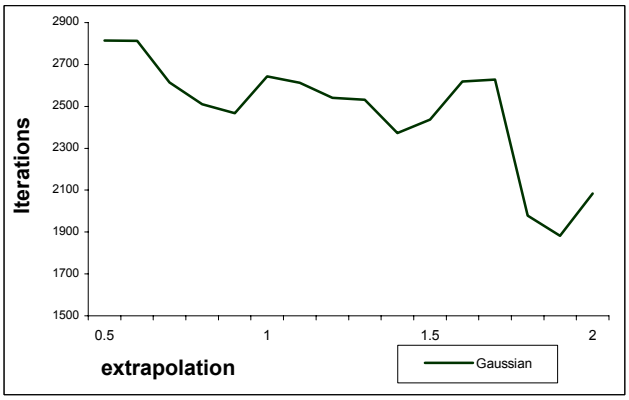

Fig 7: Effect of extrapolation on fish data set for Gaussian kernel with σ=10

Overall, values of $\zeta > 1$ give better improvement to the speed of the algorithm. In this region, the algorithm is in *extrapolation* and as long as ζ satisfies the upper bound the algorithm converges. We found that for most cases $\zeta \geq 2$ resulted in oscillation and non-convergence. In the case of the fish dataset, the upper bound for ζ was around 1.95 for a Gaussian kernel with a kernel width of 10, before the algorithm stopped converging. On closer inspection, we found that a kernel width of this size still resulted in a highly inseparable feature space with many Lagrange multipliers at bound. This leads us to believe that extrapolation works better when the optimal solution has more variables at the boundary. However, we have a problem of quantifying this due to the random nature of the working set choice by Platt.

The introduction of the cache to store Support Vectors presents this randomness, which affects the working set choice, and hence also our empirical results. In the original SMO algorithm, Platt stored the errors of the training examples that had Lagrange Multipliers, which were Support Vectors. In our algorithm, our cache imposes an ordering on the Support Vectors obtained at each epoch of the iteration. We observed from Fig 4 empirically that sorting the Support Vectors to be updated according to some criteria in SMO does vary the

speed as compared to a random selection of Support Vectors. This result requires us to do several runs to average out the effect imposed by sorting in the cache. We then ran a small experiment to remove the effect of this random choice of working sets. This was done on the UCI adult 1 dataset over a range of C and the results are reported in Fig 9-Fig 11. It can be seen that extrapolation works better for larger values of C. Geometrically, as C increases the size of the feasible region increases while the Newtonian step remains the same. Hence during the initial stages of the iteration, we would like to determine the variables at bound quickly and the addition of the extrapolation parameter seems to boost the variables to the bound thus increasing the rate of convergence. Further empirical evidence can also be seen by reexamining Fig 6 where extrapolation does not work too well for small values of C. In fact, our view of extrapolation seems to be as a boost to the basic Newtonian step similar to our previous work on linear stationary iterative methods [10].

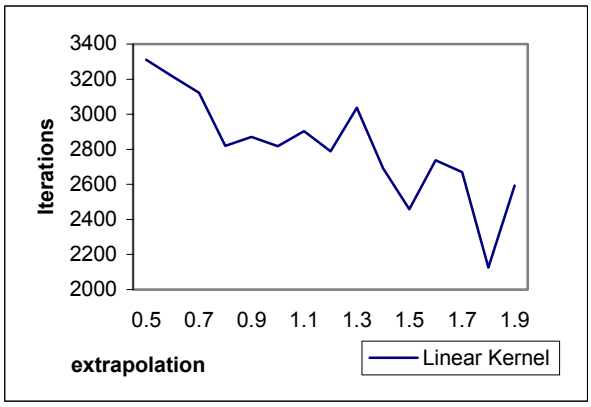

Fig 8: Effect of extrapolation on fish data set for linear kernel

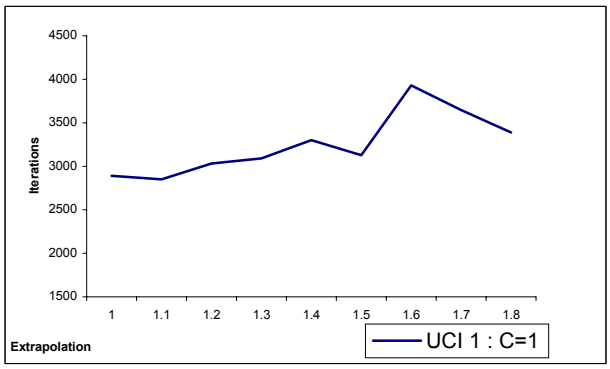

Fig 9: Extrapolation on UCI 1 without random heuristics for Gaussian kernel with C=1

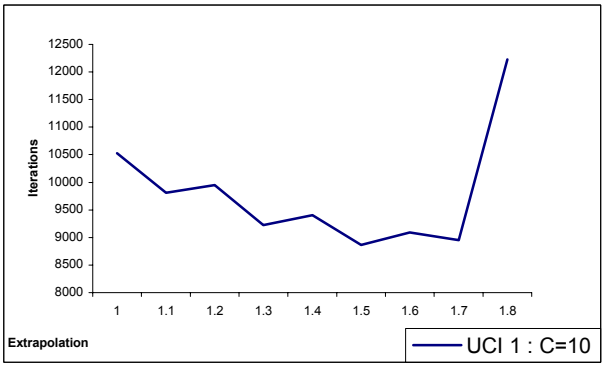

Fig 10: Extrapolation on UCI 1 without random heuristics for Gaussian kernel with C=10

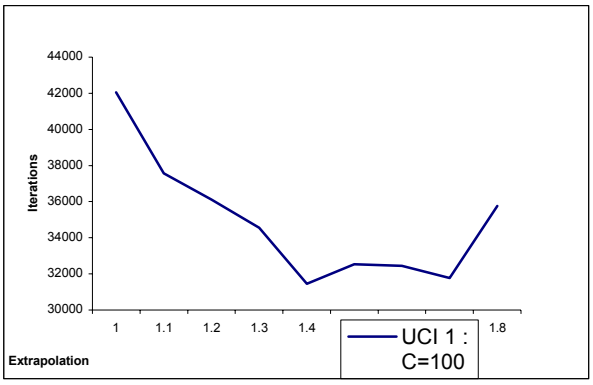

Fig 11: Extrapolation on UCI 1 without random heuristics for Gaussian kernel with C=100

The results from the experiments on the sorting indicate that the sequence in which the variables are updated do affect the rates of convergence of the decomposition algorithm. Certain orders give faster convergence while others slow the algorithm down tremendously. However, unfortunately we have yet to understand this observation clearly from a mathematical viewpoint or provide concrete justification for the optimal choice of sorting order. We do know however that, if Platt used a fixed size cache for errors, not all the Support Vectors were present in the cache at any one time, so indirectly the choice of points available for update was random and restricted to the size of the cache. In our case, we have the full array of Support Vectors to choose from and this presents some form of partial ordering on the domain of Lagrange Multipliers. We believe the analysis of the dynamics of the SMO algorithm under the choice of updating pairs would be worthwhile and more involving in order to increase its performance.

The analysis of the optimal value of extrapolation parameter ζ, is an ongoing research topic for us. In all our benchmark tests, the use of ζ results in an average improvement of about 35% compared to the original SMO without extrapolation. Intuitively, the largest value of ζ is 2, which geometrically means double the step size and anything larger would result in us not taking a step to the minimum of the objective function. Finer values of ζ were observed to give even better algorithm performance. This observation underscores the need for a theory to define the optimal value of ζ and relate it in some way to the kernel matrix. We have given the necessary condition for ζ to ensure that the iteration strictly reduces the objective function, however we do not know for sure yet whether this condition is sufficient to ensure the convergence of the algorithm. We believe however that the decomposition method is convergent as long as the basic update rule used is norm reducing on the objective function. Lin [11] has given a convergence proof for [7] in terms of a counting method, but we intend to relate the norm-reduction properties of the iteration to the convergence of the decomposition method.

It should be noted here that a consequence of the above observation is that we could use a different update rule instead of the Newton rule and apply the decomposition method to it. We can then investigate the optimal update rule in the decomposition setting with the dynamic determination of the threshold b. i.e. always maintaining the validity of the equality constraint or apply it to formulations that do not require explicit preservation of the equality constraint e.g.[10]. We believe that different working sets require different degrees or values of extrapolation hence the possibility of an adaptive extrapolation algorithm. The problem with this is that computing the optimal extrapolation parameter at each iterative step may add unnecessary overhead to the iteration and degrade the algorithm speed instead. We plan to look into this possibility in the near future.

VI. CONCLUSION

We have proposed applying extrapolation and observed some improvement to the rates of convergence with better improvements for difficult problems i.e. larger values of C. We believe that the speed of this algorithm can be further improved if we apply newer methods that improve the computation of the threshold in SMO. Our future work will concentrate on investigating the proper choice of working set pairs and the theoretical rates of convergence using an extrapolated Newtonian update rule. In particular, we will investigate the relationship between optimal values of the extrapolation parameter and the structure of the feasible region formed by the constraint set.

VII. APPENDIX

A. *Gateux-differentiable functions*

Definition VII.1 (Ortega[8])

A mapping $F : D \subset R^n \to R^m$ is G-differentiable at an interior point of D if there is some linear operator, $A \in L\left(R^n, R^m\right)$ so that for any $h \in R^n$,

$$\lim_{t \to 0} \frac{1}{t}\left(F\left(x+th\right) - Fx - tAh\right) = 0 \qquad \bullet$$

REFERENCES

[1] V. N. Vapnik, *The nature of statistical learning theory*, 2nd ed. New York: Springer, 2000.

[2] J. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," in *Advances in Kernel Methods-Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds.: Cambridge MIT Press, 1998, pp. 185-208.

[3] C.-C. Chang, C.-W. Hsu, and C.-J. Lin, "The analysis of decomposition methods for support vector machines," *Neural Networks, IEEE Transactions on*, vol. 11, pp. 1003-1008, 2000.

[4] S. K. S. S.S. Keerthi, C. Bhattacharyya and K.R.K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design,," Control Division, Dept. of Mechanical Engineering,National University of Singapore CD-99-14, 1999.

[5] C.-J. Lin, "LIBSVM," http://www.csie.ntu.edu.tw/~cjlin/.

[6] L. A. Hageman and D. M. Young, *Applied iterative methods*. New York: Academic Press, 1981.

[7] T. Joachims, "Making Large Scale Support Vector Machine Learning Practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds.: Cambridge , MIT Press, 1998, pp. 169-184.

[8] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. New York,: Academic Press, 1970.

[9] C. L. M. Blake, C.J., "UCI Repository of machine learning databases Irvine, CA," University of California, Department of Information and Computer Science, 1998.

[10] D. Lai, M. Palaniswami, and N. Mani, "Fast linear stationary methods for automatically biased support vector machines," presented at Neural Networks, 2003. Proceedings of the International Joint Conference on, 2003.

[11] C.-J. Lin, "On the convergence of the decomposition method for support vector machines," *Neural Networks, IEEE Transactions on*, vol. 12, pp. 1288-1298, 2001.

[12] C.-J. Lin, "Linear convergence for a decomposition method for Support Vector Machines," November 2001.