On Extending the SMO Algorithm Sub-Problem

Christopher Sentelle, Michael Georgiopoulos, Georgios C. Anagnostopoulos, and Cynthia Young

***Abstract*—The Support Vector Machine is a widely employed machine learning model due to its repeatedly demonstrated superior generalization performance. The Sequential Minimal Optimization (SMO) algorithm is one of the most popular SVM training approaches. SMO is fast, as well as easy to implement; however, it has a limited working set size (2 points only). Faster training times can result if the working set size can be increased without significantly increasing the computational complexity. In this paper, we extend the 2-point SMO formulation to a 4-point formulation and address the theoretical issues associated with such an extension. We show that modifying the SMO algorithm to increase the working set size is beneficial in terms of the number of iterations required for convergence, and shows promise for reducing the overall training time.**

I. Introduction

The Support Vector Machine is a classification model, maintaining excellent generalization capabilities along with a built-in resistance to overtraining. This generalization performance is based upon strong theoretical foundations first introduced by Vapnik [1]. Instead of performing empirical risk minimization, as many other models do, the SVM algorithm performs structural risk minimization. This simply means that the algorithm seeks a balance between fitting function complexity and training error. The SVM algorithm works by finding a separating hyperplane between two classes of data that maximizes the margin between the closest data point and the separating hyperplane. Vapnik shows that this is equivalent to structural risk minimization. The primal, soft-margin problem formulation is shown in (1).

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ &\text{s.t.} \quad y_i\left[\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right] - 1 + \xi_i \geq 0 \end{aligned} \tag{1}$$

Manuscript received January 31, 2007. This work was supported in part by the NSF grants: CRCD: 0203446, CCLI: 0341601, DUE: 05254209, IIS-REU: 0647120, and IIS-REU-0647120.

C. Sentelle is with the University of Central Florida, Orlando, FL 32816 USA. He is now with the School of Electrical Engineering and Computer Science. (e-mail: csentelle@cfl.rr.com) .

M. Georgiopoulos is with the University of Central Florida, Orlando, FL 32816 USA. He is now with the School of Electrical Engineering and Computer Science. (phone:407-823-5338; e-mail: michaelg@mail.ucf.edu).

G. C. Anagnostopoulos is with the Florida Institute of Technology, Melbourne, FL 32901 USA. He is now with the Department of Electrical and Computer Engineering. (email: georgio@fit.edu).

C. Young is with the University of Central Florida, Orlando, FL 32816 USA. She is now with the Department of Mathematics. (email: cyoung@mail.ucf.edu).

where, $\mathbf{w}$, is a vector normal to the separating hyperplane, $1/\|\mathbf{w}\|$, is the width of the margin, $y_i \in \{1, -1\}$, is the label of each data point, $\mathbf{x}_i$, is the i^{th} data point, and b, is a threshold which defines the offset of the separating hyperplane from the origin. A slack variable, ξ_i, relaxes the constraint for non-linearly separable data points and a penalty term $C\sum_{i=1}^{n}\xi_i$ is added to the minimization problem, which limits the number of data points violating the margin constraint. Equation (2) shows the primal problem formulation rewritten in its dual form.

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \\ &\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \end{aligned} \tag{2}$$

where, α_i, is the Lagrange multiplier associated with each data point and $K\left(\mathbf{x}_i, \mathbf{x}_j\right)$ is a non-linear mapping of the dot-product $\langle x_i, x_j\rangle$ or kernel function. Equation (3) shows this rewritten in matrix form.

$$\begin{aligned} &\text{maximize} \quad \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} \\ &\text{s.t.} \quad \boldsymbol{\alpha}^T\mathbf{y} = 0, \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1} \end{aligned} \tag{3}$$

where $\mathbf{Q}$ is defined as $(\mathbf{Q})_{ij} = y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right)$ and $\mathbf{1}$ is a vector of ones. This is a quadratic programming problem with a single equality constraint and two inequality constraints per input vector, $\mathbf{x}_i$.

This quadratic problem becomes intractable as the number of data points becomes large due to the associated number of inequality constraints and the size of the kernel matrix, $\mathbf{Q}$. Vapnik [1] first introduced the idea of chunking where more manageable sub-problems are solved. These sub-problems consisted of the worst Karush-Kuhn-Tucker (KKT) violators and data points having non-zero alpha values. Unfortunately, the sub-problem would vary in size as convergence progressed. Osuna *et al.* [2], then, introduced the notion of using fixed size sub-problems where at least one KKT violator was added to the sub-problem at each iteration. Of course convergence was still slow leading to a search for more advanced training algorithms.

Later, the Sequential Minimal Optimization (SMO) algorithm [3] was introduced. The SMO algorithm implemented decomposition to an extreme where only two input vectors are selected, at each iteration. In previous chunking algorithms a quadratic programming technique such as an interior point method is applied to each sub-problem, whereas, with SMO, Platt derives an analytical expression for solving the sub-problem. Platt's algorithm is easy to implement and is found to significantly outperform existing algorithms [3]. In this paper, we extend the SMO idea, where 2 points are updated at a time, to the case where 4 points are updated.

The organization of the paper is as follows. Section II discusses, in detail, the literature review of approaches that have been introduced to solve (efficiently) the quadratic optimization problem with constraints, associated with the SVM formulation (see equation (3)), and the motivation behind considering a 4-point SMO. In Section III, theoretical issues regarding the 4-point SMO formulation are dealt with, such as the update of the 4 variables, the degeneracies that one might encounter in the 4-point SMO formulation, and the method to select the 4-points that are to be updated. Section IV contains preliminary experiments that we have performed with the 4-point SMO algorithm and comparisons with a state-of-the art SVM training algorithm implementation, LIBSVM. These results indicate that there is merit in considering a 4-point SMO approach to solve the SVM problem. Finally, in Section V we present a summary of our work, and some conclusive remarks.

II. LITERATURE REVIEW AND MOTIVATION

There are a number of approaches that have been introduced in the literature to solve the SVM problem, in addition to the aforementioned SMO algorithm. For instance, Joachims, introduced in [4], the concept of performing decomposition for any number of even data points and employs an interior point method to solve each sub-problem. In addition, Joachims introduced a novel method for selecting data points for each sub-problem that is linear in time, introduced the concept of shrinking, and implemented kernel caching, which Platt mentioned in his initial publication but did not implement. Joachims [4] reports that the SVMLight software, based upon these concepts, is faster than the SMO algorithm.

Recognizing some issues with Platt's algorithm, a modification to the data point selection and termination criterion were suggested by Keerthi [5]. Keerthi shows that the Platt algorithm could perform more iterations than necessary when using a single threshold, and, therefore, introduces the notion of using a dual threshold. Platt's algorithm also maintains inefficiencies in the manner in which it selects input vectors for the sub-problem since, if the heuristic for selecting a second input vector fails, a random mechanism is employed. Keerthi's improvement eliminated this randomness.

While Keerthi's algorithm was shown to outperform Platt's original implementation of SMO, Fan *et al.* [6] introduced yet another modification to the SMO algorithm for selecting input vectors. It is shown that Keerthi's point selection algorithm is the Working Set Selection (WSS) introduced by Joachims when two points are selected [6]. The SVMLight algorithm approximates the objective function using a 1st order Taylor's series approximation and builds a linear programming problem from this for selecting data points for the next iteration. On the other hand, Fan *et al.* show that a 2nd order Taylor's series approximation of the objective function may yield significantly better results when applied to the SMO algorithm. Fan, *et al.* [6] report significant convergence timing improvements based upon their working set selection using second order information.

In addition to the mainstream SVM training algorithms, there have been a host of other modifications introduced such as the LS-SVM [7], LSVM [8], SSVM [9], and RSVM [10], to name a few. In these algorithms, the authors reformulate the primal problem associated with SVM and/or perform techniques for working with smaller subsets of the data. In each case, the algorithm is either not competitive for the non-linear kernel (LSVM), the support vectors are no longer sparse (LS-SVM) or training time is traded for some small reduction in accuracy (RSVM) [11].

Our motivation for considering a 4-point SMO algorithm stems from the realization that some problems might benefit from considering more than two points per iteration from the standpoint of faster convergence to the solution. For example, although the SVMLight methodology is much more difficult to implement, it affords the possibility of quicker convergence by considering more points per iteration. In fact, in two of the data sets identified by Joachims [4], the Ohsumed data set and the Baluja face image data set, the optimal number of data points per iteration was identified to be 20, while in other cases; the optimal working set size was 2. The current SMO implementation does not provide the flexibility of increasing the working set selection size beyond two, despite the fact that there may be potential practical advantages for doing so. In this paper, we explore the possibility of extending the SMO algorithm to consider more than 2 points per iteration while maintaining the ease of implementation associated with SMO. The theoretical issues of such an extension are addressed in Section III, while preliminary experimental results are provided in Section IV, justifying the practical merit of such an investigation.

III. 4-POINT SMO

To derive the 4-point SMO formulation, we first start with (3). The equality constraint is ensured at each update using 4 points by satisfying (4).

$$\begin{aligned}&\alpha_1 y_1+\alpha_2 y_2+\alpha_3 y_3+\alpha_4 y_4 = \\ &\quad \alpha_1^{old} y_1+\alpha_2^{old} y_2+\alpha_3^{old} y_3+\alpha_4^{old} y_4\end{aligned} \tag{4}$$

Using a Lagrange multiplier and adding to the original objective function, we obtain

$$L(\boldsymbol{\alpha},r)=\mathbf{1}^T\boldsymbol{\alpha}-\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha}-r\left(\left(\boldsymbol{\alpha}^{\mathbf{T}}-\boldsymbol{\alpha}_{\mathbf{old}}^{\mathbf{T}}\right)\begin{bmatrix}\mathbf{y_S}\\ \mathbf{0}\end{bmatrix}\right) \tag{5}$$

The gradient of the objective function, with respect to $\boldsymbol{\alpha}$, is

$$\nabla L(\boldsymbol{\alpha},r)=\mathbf{1}^T-\mathbf{Q}\boldsymbol{\alpha}-r\begin{bmatrix}\mathbf{y_S}\\ \mathbf{0}\end{bmatrix} \tag{6}$$

We are only interested in $\frac{\partial L}{\partial\alpha_1},\frac{\partial L}{\partial\alpha_2},\frac{\partial L}{\partial\alpha_3},\frac{\partial L}{\partial\alpha_4}$ which can be obtained from (6) by extracting the first four rows of the gradient, as seen in (7).

$$\begin{aligned}\frac{\partial L}{\partial\alpha_1}&=1-y_1\sum_{i=1}^{n}\alpha_i y_i k_{1i}-ry_1\\ \frac{\partial L}{\partial\alpha_2}&=1-y_2\sum_{i=1}^{n}\alpha_i y_i k_{2i}-ry_2\\ \frac{\partial L}{\partial\alpha_3}&=1-y_3\sum_{i=1}^{n}\alpha_i y_i k_{3i}-ry_3\\ \frac{\partial L}{\partial\alpha_4}&=1-y_4\sum_{i=1}^{n}\alpha_i y_i k_{4i}-ry_4\end{aligned} \tag{7}$$

If we define $v_i=\sum_{j=5}^{n}\alpha_j y_j k_{ij}$, solve each expression of (7) in terms of $\alpha_1,\alpha_2,\alpha_3,$ and α_4 and rewrite these in a matrix form, we obtain

$$\begin{bmatrix}\mathbf{Q}_s & \mathbf{y}_s\\ \mathbf{y}_s^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_s\\ r\end{bmatrix}=\begin{bmatrix}\mathbf{1}-\mathbf{u}\\ \mathbf{y}^T\boldsymbol{\alpha}_s^{old}\end{bmatrix} \tag{8}$$

where $(\mathbf{u})_i=y_i v_{i,}\ i\in\{1..4\}$, $\mathbf{Q}_s$, is a sub-matrix of $\mathbf{Q}$ containing the kernel matrix for the 4 selected points, $\mathbf{y}_s$ contains the labels of the 4 points, and $\boldsymbol{\alpha}_s$ contains the alpha values for the 4 data points.

By solving this equation, we obtain the unconstrained 4-point sub-problem; however, we must also consider the inequality constraints associated with the original problem (3). In the case of more than two points, the problem becomes one of constraining the solution on the surface of a hyperplane based upon the inequality constraints defined, now, by the interior of a hypercube. The computation entailed in considering all possible combinations with more than 2 points grows exponentially. There are a total of 2^{n-1} possible hyperplane orientations based upon the labels, y_i, where the sign of the normal to the hyperplane is ignored. For four data points, $n=4$, there are 8 relevant orientations of the hyperplane. Each hyperplane orientation can intersect the hypercube in multiple ways. In addition, it is discovered that the intersections are no longer points, as in the SMO case, but, they are planes. Therefore, it seems that the analytical methodology suggested by Platt for clipping the data points is no longer applicable.

Therefore, the Theil-Van de Panne procedure [12] is introduced to solve this problem. This procedure provides a mechanism for searching the inequality constraints to find those that are active in the final solution. The author in [12] introduces the quadratic programming problem

$$\begin{aligned}&\text{maximize}\quad \psi(\mathbf{x})=\mathbf{a}'\mathbf{x}-\frac{1}{2}\mathbf{x}'\mathbf{B}\mathbf{x}\\ &\text{s. t.}\quad \mathbf{C}'\mathbf{x}\le\mathbf{d}\end{aligned} \tag{9}$$

where $\mathbf{B}$ is a positive definite matrix of size $n\times n$ and $\mathbf{C}$ is a matrix of the size $n\times m$, with m constraints. A subset of the active constraints is referred to as S, and the solution where the constraints, S, are active is referred to as $\mathbf{x}_S$, which is both feasible $(\mathbf{C}'\mathbf{x}_S\le\mathbf{d})$ and optimal.

In general, the procedure works as follows. First, the solution to the unconstrained problem, $\mathbf{x}^{\phi}$, is solved. If this solution is feasible, then this is also the optimal solution. However, if there are violated constraints, then *at least one* of the constraints violated will be an active constraint in the final solution. The heuristic, then, considers each violator of $\mathbf{x}^{\phi}$, and includes each violator, in turn, as an active constraint and re-solves the optimization problem to obtain $\mathbf{x}^{S/1}$ (where $\mathbf{x}^{S/f}$ indicates a set of active constraints, S, containing f elements). In this case, $\mathbf{x}^{S/1}$ implies we are considering all solutions with one active constraint. Each of these solutions, may, in turn, identify additional violators. If any solution $\mathbf{x}^{S/1}$ is feasible, then that is an optimal solution and the algorithm has completed. If there are no feasible $\mathbf{x}^{S/1}$ solutions, then the algorithm considers all solutions $\mathbf{x}^{S/2}$, that is, all solutions containing two active constraints. If a solution, $\mathbf{x}^{S/2}$, is found to be feasible, we must consider the Lagrange multiplier associated with that solution to determine if it is not only feasible but optimal. In fact, there could be more than one $\mathbf{x}^{S/2}$ feasible solution, but only one is optimal in terms of providing the maximal value of the objective function. This process continues until the optimal solution is found. In our implementation, the original matrix equation can be augmented in order to solve for the solution to $\mathbf{x}^S$ as shown in (10) where α_1 is constrained to the value C.

$$\begin{bmatrix} \mathbf{Q}_s & \mathbf{y}_s & \mathbf{e}_1 \\ \mathbf{y}_s^T & 0 & 0 \\ \mathbf{e}_1^\mathbf{T} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_s \\ r \\ u \end{bmatrix} = \begin{bmatrix} \mathbf{1}-\mathbf{u} \\ \mathbf{y}^T \boldsymbol{\alpha}_s^{old} \\ C \end{bmatrix} \tag{10}$$

$$\text{where, } \mathbf{e_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Any number of constraints can be added to this equation by simply augmenting the matrix for each constraint. Note that the resulting matrix on the left hand-side of the equation remains symmetric and sparse.

A. Solutions for Non-Degenerate Cases

In the non-degenerate case, a single, global maximum exists and $\mathbf{Q}_s$ is of full rank. Note that (8) contains a symmetric, partitioned matrix and a closed form solution exists.

Let $\mathbf{Q}_s$ be a positive definite, symmetric matrix, and $\mathbf{A}$ the following partitioned matrix,

$$\mathbf{A} \mathrel{\hat{=}} \begin{bmatrix} \mathbf{Q}_s & \mathbf{y} \\ \mathbf{y}^T & 0 \end{bmatrix}. \tag{11}$$

Then, the inverse of $\mathbf{A}$ exists and is given as,

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{Q}_s & \mathbf{y} \\ \mathbf{y}^T & 0 \end{bmatrix}^{-1} = \frac{1}{g} \begin{bmatrix} g\mathbf{Q}_s{}^{-1} + \mathbf{Q}_s{}^{-1}\mathbf{y}\mathbf{y}^T\mathbf{Q}_s{}^{-1} & -\mathbf{Q}_s{}^{-1}\mathbf{y} \\ -\mathbf{y}^T\mathbf{Q}_s{}^{-1} & 1 \end{bmatrix} \tag{12}$$

$$g \mathrel{\hat{=}} -\mathbf{y}^T\mathbf{Q}_s{}^{-1}\mathbf{y}$$

since $\mathbf{Q}_s$ is invertible and $\mathbf{y}^T\mathbf{Q}_s{}^{-1}\mathbf{y} \neq 0$ due to the positive definiteness of $\mathbf{Q}_s$. Using this, the solution to $\boldsymbol{\alpha}$ is computed as

$$\mathbf{A}\begin{bmatrix} \boldsymbol{\alpha} \\ u \end{bmatrix} = \mathbf{b} \quad \Leftrightarrow \quad \boldsymbol{\alpha} = \frac{1}{g}\left[\left(g\mathbf{I} + \mathbf{Q}_s{}^{-1}\mathbf{y}\mathbf{y}^T\right)\mathbf{Q}_s{}^{-1} \quad -\mathbf{Q}_s{}^{-1}\mathbf{y}\right]\mathbf{b}.$$

The closed form solution affords the flexibility of incremental computation as constraints are added, when employing the Theil-Van de Panne procedure. In this case, we can augment the $\mathbf{A}$ matrix as

$$\mathbf{A}_1 \mathrel{\hat{=}} \begin{bmatrix} \mathbf{A} & \mathbf{e}_i \\ \mathbf{e}_i^T & 0 \end{bmatrix} \tag{13}$$

and compute the inverse as follows

$$\mathbf{A}_1^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{e}_i \\ \mathbf{e}_i^T & 0 \end{bmatrix}^{-1} = \frac{1}{g} \begin{bmatrix} g\mathbf{A}^{-1} + \left[\mathbf{A}^{-1}\right]_{.,i}\left[\mathbf{A}^{-1}\right]_{i,.} & -\left[\mathbf{A}^{-1}\right]_{.,i} \\ -\left[\mathbf{A}^{-1}\right]_{i,.} & 1 \end{bmatrix} \tag{14}$$

$$g \mathrel{\hat{=}} -\mathbf{e}_i^T\mathbf{A}^{-1}\mathbf{e}_i = -\left[\mathbf{A}^{-1}\right]_{i,i}$$

where $\mathbf{e_i}$ is a unit vector with a one in the ith position representing the augmented equality constraint, and $\left[\mathbf{A}^{-1}\right]_{i,.}$, $\left[\mathbf{A}^{-1}\right]_{.,i}$, $\left[\mathbf{A}^{-1}\right]_{i,i}$ denote the ith row, ith column and ith diagonal element of matrix $\mathbf{A}^{-1}$. Furthermore, replacing $\mathbf{A}$ with $\mathbf{A_1}$ and $\mathbf{e}_i$ with another unit vector, we can compute, recursively, the inverse of an augmented matrix $\mathbf{A_2}$ in a similar spirit to (14). As a result, the Theil Van de Panne procedure can be performed in an incremental fashion resulting in far fewer computations than would be required otherwise.

B. Solutions for Degenerate Cases

Degeneracy occurs when $\mathbf{Q}_s$ is not invertible, in the presence of duplicate data points or when the dimensionality of the input vectors is less than 4 and a linear kernel is employed for the 4-point SMO. A degeneracy will also occur when $\mathbf{Q}_s$ is not positive definite. There are two methods for dealing with the degeneracies. The appropriate rows and columns of (8) can be zeroed out to convert the problem to a 3 or 2-point SMO update, or, in the case of certain degeneracies, an approach can be taken where all alpha values are assumed to violate the constraints and the Theil Van de Panne procedure is employed to find the optimal, feasible solution set. The details are omitted due to lack of space.

C. Working Set Selection

In Fan *et al*. [6], the following Working Set Selection (WSS) heuristic is used to select a pair of data points for the next iteration.

Select

$$\begin{aligned} i &\in \arg\max_t \left\{ -y_t \nabla f\left(\boldsymbol{\alpha}^k\right)_t \,\middle|\, t \in I_{up}\left(\boldsymbol{\alpha}^k\right) \right\}, \\ j &\in \arg\min_t \left\{ -\frac{b_{it}^2}{a_{it}} \,\middle|\, \begin{array}{l} t \in I_{low}\left(\boldsymbol{\alpha}^k\right), \\ -y_t \nabla f\left(\boldsymbol{\alpha}^k\right)_t < -y_i \nabla f\left(\boldsymbol{\alpha}^k\right)_i \end{array} \right\} \end{aligned} \tag{15}$$

where

$$I_{up}(\boldsymbol{\alpha}) \triangleq \{t \mid \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\},$$
$$I_{low}(\boldsymbol{\alpha}) \triangleq \{t \mid \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\} \quad (16)$$

$$b_{it} = -y_i \nabla f(\boldsymbol{\alpha}^k)_i + -y_t \nabla f(\boldsymbol{\alpha}^k)_t,$$
$$a_{it} = K_{ii} + K_{tt} - 2K_{it} \quad (17)$$

$\nabla f(\boldsymbol{\alpha}^k)$ is the gradient, y_i is the label, and K_{ij} is the kernel function applied to input vectors x_i and x_j. To deal with the possibility of non-positive definite kernels, the author introduces the idea of setting $a_{it} = \tau$ in the case when $a_{it} < 0$ where τ is a user selectable parameter normally set to 1e-12. It can be shown, similar to the theory of the 2-point case, that the algorithm depicted in (18) and (19) can be applied in the 4-point case and will yield an optimal update per step.

Select

$$s \in \arg\max_i \left\{-y_i \nabla f(\boldsymbol{\alpha}^k)_i \,\middle|\, i \in I_{up}(\boldsymbol{\alpha}^k)\right\},$$
$$t \in \arg\min_i \left\{-\frac{b_{it}^2}{a_{it}} \,\middle|\, \begin{array}{l} i \in I_{low}(\boldsymbol{\alpha}^k), i \notin \{u\}, \\ -y_i \nabla f(\boldsymbol{\alpha}^k)_i < -y_t \nabla f(\boldsymbol{\alpha}^k)_t \end{array}\right\} \quad (18)$$

$$u \in \arg\min_i \left\{-y_i \nabla f(\boldsymbol{\alpha}^k)_i \,\middle|\, i \in I_{low}(\boldsymbol{\alpha}^k)\right\},$$
$$v \in \arg\min_i \left\{-\frac{b_{vi}^2}{a_{it}} \,\middle|\, \begin{array}{l} i \in I_{up}(\boldsymbol{\alpha}^k), i \notin \{s,t\}, \\ -y_v \nabla f(\boldsymbol{\alpha}^k)_v < -y_i \nabla f(\boldsymbol{\alpha}^k)_i \end{array}\right\} \quad (19)$$

It was found that as the algorithm gets closer to the optimal solution there were cases where two sets of data points could not be found without duplication. When this occurs, both of the initial data points s,u can be chosen from either I_{up} or from I_{low}.

IV. RESULTS

To illustrate the merit of the 4-point SMO formulation, we perform some preliminary experiments using the data sets Sonar, German, and Four Class. The German dataset is from the Statlog collection [13]. The Four Class problem is from [14] and was transformed to a two-class set. Finally, Sonar was obtained from the UCI repository [15]. All data sets were acquired from [16] where they have been formatted for use with LIBSVM. The linear kernel and RBF kernels were employed with $\log_2 C$ varying from -3 to 9 in steps of 2, for the linear kernel, and with $\log_2 C$ varying from -5 to 13 in steps of 2, $\log_2 \gamma$ varying from -15 to 3 in steps of 2, for the RBF kernel. The same stopping criterion is employed for both LIBSVM and the 4-Point SMO with the maximum distance from the optimal solution specified as 10^{-3}. We chose to compare our 4-point SMO algorithm with the latest version of LIBSVM (ver. 2.82) [16] compiled as a MATLAB MEX file. LIBSVM implements the algorithm as described by Fan *et al.* [6]. The 4-point SMO algorithm was also implemented as a MATLAB MEX file. In all cases, kernel caching and shrinking was disabled, as this feature is not currently implemented in the 4-Point SMO algorithm.

TABLE I
COMPARISON OF THE 4-POINT SMO AND LIBSVM TRAINING ALGORITHMS.

Data Set	LIBSVM	4-point SMO
	Time (sec) Iterations	Time (sec) Iterations
Four Class (RBF)	134.97 476,014	74.806 107,441
Sonar (RBF)	11.41 63,293	11.42 33,316
Sonar (linear)	10.51 80,003	8.05 45,330
German (RBF)	578.74 958,260	671.56 587,297
German (linear)	622.36 1,965,342	466.59 1,153,402

The measure of comparison between the algorithms was the total training time required for all combinations of C and γ for a specific problem (dataset). This seems reasonable since most practical applications will require evaluation of these parameters to find the optimal settings for C and γ. Classification performance, the number of support vectors, and threshold, not reported here, were compared to ensure neither algorithm terminated prematurely. Table I depicts the total training times and number of iterations required for convergence by each algorithm.

The time per iteration for both LIBSVM and the 4-point SMO is dominated by computation of the error cache update. The LIBSVM algorithm requires $2N$ kernel computations at each iteration since two alpha values are updated while the 4-point SMO requires $4N$ kernel computations. As a result, we expect that the 4-point SMO algorithm must reduce the number of iterations by ½ compared to the LIBSVM algorithm in order to achieve comparable training times. In fact, we see this for the Sonar dataset, with the RBF kernel, where the number of iterations required by the 4-Point SMO algorithm is nearly ½ that of the LIBSVM training algorithm and training times are comparable.

In summary, we observe that the 4-Point SMO consistently converges in fewer iterations than the LIBSVM algorithm. In addition, a comparison of training time reveals that the 4-Point SMO algorithm is faster when compared to

LIBSVM for the Four Class data set, Sonar dataset with a linear kernel, and the German dataset with a linear kernel. In fact, for the Four Class dataset, the iterations required by the 4-Point SMO algorithm are approximately ¼ that of LIBSVM algorithm and training time is nearly ½. We note a worst case increase of the training time for the German dataset with the RBF kernel where there is a 16% increase in training time. Based on the results reported by Joachims [4], we predict that increasing the working set size may not be beneficial for all data sets, as observed by the German dataset in Table I.

V. SUMMARY AND CONCLUSIONS

In summary, we have presented an extension to the SMO algorithm for larger working sets (4 points in this case). We have shown that extending the SMO working set size consistently decreases the number of iterations required for convergence and, in some cases, results in a decrease in training time. In terms of efficiency, the computation time for both the 2-point SMO and the 4-point extension is dominated by the update of the error cache. As a result, the time per iteration is roughly doubled for the 4-point SMO algorithm. A decrease in training time will occur if the number of iterations required for convergence can be substantially reduced by increasing the working set size. The promising results, presented here, appear to merit continued research into improving the 4-point SMO algorithm and further increasing the working set size to discover the point at which computational complexity prevents further reduction in total training time.

Future work includes finding more efficient methods for dealing with the non-invertible sub-problem kernel matrix, by researching alternatives to the Theil-Van de Panne procedure, and improved working set selection techniques based upon the WSS introduced by Fan *et al.*, all of which could lead to a faster implementation of the 4-point SMO algorithm. In addition, it seems worthwhile to expand the ideas, presented here, to an arbitrary N-point SMO where the efficiency, in some cases, might be expected to further increase.

ACKNOWLEDGMENT

Christopher Sentelle gratefully acknowledges the support of the College of Engineering & Computer Science and the I2Lab at the University of Central Florida.

REFERENCES

[1] V. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, (1982).

[2] E. Osuna, R. Freund, and F. Girosi, “An improved training algorithm for support vector machines,” *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*, pp. 276 – 285, New York, 1997a. IEEE.

[3] J. C. Platt, “Fast Training of Support Vector Machines Using Sequential Minimal Optimization,” *Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[4] T. Joachims, “Making large-Scale SVM Learning Practical. Advances in Kernel Methods,” *Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[5] S. S. Keerthi, "Improvements to Platt's SMO algorithm for SVM classifier design", *Neural Computation*, vol. 13, 2001, pp 637-649.

[6] R.-E. Fan, P.-H. Chen, and C.-J. Lin. “Working set selection using the second order information for training SVM.” *Journal of Machine Learning Research*, vol. 6, 1889-1918, 2005.

[7] J. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural Processing Letters*, vol. 9: 293–300, 1999.

[8] O. L. Mangasarian and D. R. Musicant, “Lagrangian support vector machines,” *J. Machine Learning Res.*, vol. 1, pp. 161–177, 2001.

[9] Y. Lee and O. L. Mangasarian, “SSVM: a smooth support vector machine for classification,” *Computational Optimization and Applications*, vol. 20, no. 1, pp. 5-22, 2001.

[10] Y.-J. Lee and O. L. Mangasarian, “RSVM: reduced support vector machines,” in *Proc. 1st SIAM Int. Conf. Data Mining*, 2001.

[11] K.-M. Lin, C.-J. Lin, “A study on reduced support vector machines,” *IEEE Transactions on Neural Networks*, vol. 14 (6) (2003) pp. 1449 - 1459.

[12] J. C. G. Boot, *Quadratic Programming*, Vol. 2, Rand McNally & Company, pp. 95-124, 1964.

[13] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J., 1994. Data available at http://www.ncc.up.pt/liacc/ML/statlog/datasets.html.

[14] T. K. Ho and E. M. Kleinberg, “Building projectable classifiers of arbitrary complexity.” *Proceedings of the 13th International Conference on Pattern Recognition*, pp. 880–885, Vienna, Austria, August 1996.

[15] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, (1998). *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[16] C. Chang and C. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.