An Improved Algorithm for the Solution of the Entire Regulation Path of Support Vector Machine

*Chong-Jin Ong, Shi-Yun Shao, Jian-Bo Yang

Abstract

This paper describes an improved algorithm for the numerical solution to the Support Vector Machine (SVM) classification problem for all values of the regularization parameter, C. The algorithm is motivated by the work of Hastie *et. al.* and follows the main idea of tracking the optimality conditions of the SVM solution for descending value of C. It differs from Hastie's approach in that the tracked path is not assumed to be one-dimensional. Instead, a multi-dimensional feasible space for the optimality condition is used to solve the tracking problem. Such a treatment allows the algorithm to handle data set having duplicate points, nearly duplicate points and linearly dependent points, a common feature among many real-world data sets. Other contributions of this paper include a unifying formulation of the tracking process in the form of a linear program, an initialization routine that deals with duplicate data points, a routine that guards against accumulation of errors resulting from the use of incremental updates, a heuristics-based routine and update formula to speed up the algorithm. The algorithm is implemented under the Matlab environment and tested with several data sets including data set having up to several thousand data points.

Index Terms

Support Vector Machine, numerical solutions of SVM, regulation path.

I. Introduction

The numerical solution of Support Vector Machine (SVM) has been the focus of much research in machine learning community in recent years. Several reliable numerical routines have since emerged, see

C.J. Ong is with the Department of Mechanical Engineering, National University of Singapore, Singapore 117576 and Singapore-MIT Alliance, J.B. Yang and S.Y. Shao are with the Department of Mechanical Engineering, National University of Singapore, Singapore, Singapore 117576 (fax: +65 67791459; Email: mpeongcj@nus.edu.sg; yangjianbo@nus.edu.sg; shaoshiyun@nus.edu.sg)

for example, [1, 2, 3, 4] and those that are designed specifically for linear kernel: [5], [6], [7], [8] and others. All of these routines solved the solution of the SVM for a particular choice of the regularization parameter, C. However, the determination of the optimal C for a data set requires the numerical routine to be invoked many times, each time with a different value of C. Hence, algorithms that provide SVM solutions for a wide range of values of C are desirable. Hastie et. al. [9] shows an approach (hereafter referred to as the Hastie's approach) on how this can be done. It is based on a one-dimensional tracking of the Karush-Kuhn-Tucker (KKT) optimality condition of the dual problem as C changes, resulting in numerical solutions for all values of C known as the SVMPath. Extensions of Hastie's approach to other problems have also appeared, see [10, 11] and references therein. Hastie's approach is an important contribution towards solving the SVMPath problem as it shows the principal steps involved. However, it has several unresolved issues. Among them, the most limiting being its inability to deal with data set having duplicate data points, nearly duplicate points or points that are linearly dependent in the kernel space. The presence of such data points is common among real-life data sets and becomes more prominent when the number of data points increases or when different kernel functions are used.

This paper shows an improvement over Hastie's approach that can deal with the difficulties mentioned above. The proposed approach is based on tracking the KKT conditions in a multi-dimensional space and allowing for non-uniqueness of the solution path. Once started, the proposed algorithm uses one single Linear Programming formulation for the tracking process. It also includes a routine to guard against accumulation of numerical errors due to incremental updates of variables, a heuristics routine and update formula to speed up the algorithm. The numerical algorithm has been tested on several data sets with duplicate points and linearly dependent points, including sets having several thousand points.

II. Review of Past Works

This section begins with a review of some well-known results related to the SVM classification problem and its numerical solutions. It also serves to set up the necessary notations needed for the discussion hereafter. Given a data set $\{x_i, y_i\}$ where $i \in \mathcal{I} := \{1, \cdots, N\}$, $x_i \in R^n$ is the feature vector and $y_i \in \{-1, +1\}$ is the corresponding label, the standard two-class classification problem for SVM takes

the form of the primal convex optimization problem (PB):

$$\min_{w,b,\xi} \quad \frac{1}{2}\, w'w + C\sum_{i=1}^{N} \zeta_i \tag{1}$$

$$y_i(w'z_i - b) \geq 1 - \zeta_i \quad \forall\ i \in \mathcal{I} \tag{2}$$

$$\zeta_i \geq 0 \quad \forall\ i \in \mathcal{I} \tag{3}$$

where $z_i := \phi(x_i)$ refers to the point in the high dimensional Hilbert space, $\mathcal{H}$, mapped into by the function $\phi : R^n \rightarrow \mathcal{H}$, w is the normal vector to the separating hyperplane $\{z|w'z - b = 1\}$ in $\mathcal{H}$ and w' is the transpose w. The standard approach to the numerical solution of PB is obtained via the dual problem (DP) in the form of

$$\min_{\alpha} \quad \frac{1}{2}\, w(\alpha)'w(\alpha) - \sum_{i=1}^{N} \alpha_i \tag{4}$$

$$0 \leq \alpha_i \leq C, \quad i \in \mathcal{I} \tag{5}$$

$$\sum \alpha_i y_i = 0 \tag{6}$$

where α_i is the Lagrange multiplier for each inequality in (2) and $w(\alpha) = \sum_{j\in\mathcal{I}} \alpha_j y_j z_j$. It is well known that DP is also a convex optimization problem and the KKT conditions (see for example [4], [12]) are constraints (5), (6) and

$$\varphi_i(\alpha, b) \geq 1 \quad \text{if } \alpha_i = 0, \tag{7}$$

$$\varphi_i(\alpha, b) = 1 \quad \text{if } 0 < \alpha_i < C, \tag{8}$$

$$\varphi_i(\alpha, b) \leq 1 \quad \text{if } \alpha_i = C, \tag{9}$$

where

$$\varphi_i(\alpha, b) := \sum_{j\in\mathcal{I}} \alpha_j k_{ji} - y_i b \text{ and } k_{ji} = y_j y_i z_j \cdot z_i. \tag{10}$$

As DP is not necessarily a strictly convex problem, the value of (α, b) satisfying (5) - (10) is not unique. The next subsection reviews the main steps of Hastie's approach, details of which can be found in [9].

A. Hastie's approach

As in Hastie's presentation, a new set of variables are defined by

$$\lambda := C^{-1}, \ \ \hat{\alpha}_i := \lambda\alpha_i, \ \forall\ i \in \mathcal{I} \text{ and } \ \ \hat{\alpha}_0 := \lambda b. \tag{11}$$

The KKT conditions of (5), (6), (7) - (9) can be rewritten in terms of these new variables as

$$0 \leq \hat{\alpha}_i \leq 1 \quad \forall\ i, \quad \sum_j \hat{\alpha}_j y_j = 0, \tag{12}$$

$$\xi_i(\hat{\alpha}, \hat{\alpha}_0, \lambda) \leq 0 \quad \text{if } i \in \mathcal{R}, \tag{13}$$

$$\xi_i(\hat{\alpha}, \hat{\alpha}_0, \lambda) = 0 \quad \text{if } i \in \mathcal{E}, \tag{14}$$

$$\xi_i(\hat{\alpha}, \hat{\alpha}_0, \lambda) \geq 0 \quad \text{if } i \in \mathcal{L} \tag{15}$$

where

$$\xi_i(\hat{\alpha}, \hat{\alpha}_0, \lambda) := 1 - \frac{\varphi_i(\hat{\alpha}, \hat{\alpha}_0)}{\lambda} \tag{16}$$

and $\mathcal{R}, \mathcal{L}, \mathcal{E}$ refer, respectively, to the index sets containing right, left and elbow points defined by

$$\mathcal{R} := \{i : \hat{\alpha}_i = 0\}, \quad \mathcal{L} := \{i : \hat{\alpha}_i = 1\} \text{ and } \quad \mathcal{E} := \{i : 0 < \hat{\alpha}_i < 1\}. \tag{17}$$

The approach of Hastie begins with an initialization phase where an appropriate set of $\hat{\alpha}$ satisfying (12) - (15) is obtained at the largest value of λ, λ^{max}. The largest λ is the value for which there is no change to $\mathcal{R}, \mathcal{L}, \mathcal{E}$ for all λ that are greater than λ^{max}. Hastie's algorithm starts by setting $\lambda^0 = \lambda^{max}$ and recursively computes $\lambda^\iota, \iota = 1, \cdots$. Each λ^ι corresponds to the value of λ where an event occurs. An event is said to have occurred when there is a change in the elements of $\mathcal{R}, \mathcal{L}$ or $\mathcal{E}$. More precisely, this happens when one or more of the following changes to the index sets take place:

- $\mathcal{E} \rightarrow \mathcal{L} \cup \mathcal{R}$: an index $i \in \mathcal{E}$ moves to $\mathcal{L}$ or $\mathcal{R}$.
- $\mathcal{L} \rightarrow \mathcal{E}$: an index $i \in \mathcal{L}$ moves to $\mathcal{E}$.
- $\mathcal{R} \rightarrow \mathcal{E}$: an index $i \in \mathcal{R}$ moves to $\mathcal{E}$.

Let the superscript ι attached to a variable or set denote its values when $\lambda = \lambda^\iota$. The main phase of Hastie's algorithm determines the value of $\lambda^{\iota+1}$ using λ^ι, $\mathcal{E}^\iota, \mathcal{L}^\iota, \mathcal{R}^\iota$ and $\hat{\alpha}^\iota$. When $\lambda^{\iota+1}$ is known, the index sets $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ and $\hat{\alpha}$ are updated accordingly based on the nature of the transition that had taken place to yield $\hat{\alpha}^{\iota+1}$, $\mathcal{E}^{\iota+1}, \mathcal{L}^{\iota+1}, \mathcal{R}^{\iota+1}$. This main phase proceeds repeatedly in increasing value of ι and decreasing value of λ^ι starting from λ^0 until termination. The determination of the value of $\lambda^{\iota+1}$ depends on the observation that (14) must hold for all λ as long as no new event happens. Using this observation and the fact that $\alpha_i, i \in \mathcal{L} \cup \mathcal{R}$ do not change in value when no new event happens, (14) can be rewritten in full as

$$\sum_{j \in \mathcal{E}} (\hat{\alpha}_j - \hat{\alpha}_j^\iota) k_{ji} - (\hat{\alpha}_0 - \hat{\alpha}_0^\iota) y_i = \lambda - \lambda^\iota \quad \forall \quad i \in \mathcal{E}. \tag{18}$$

Suppose the cardinality of $\mathcal{E}$, $|\mathcal{E}| = m$ and its elements are labelled as indices 1 to m for notational convenience. The above equation can be rewritten as

$$\hat{k}_i' \delta_{\hat{\alpha}} = \delta_\lambda \quad \forall \quad i \in \mathcal{E} \tag{19}$$

where

$$\hat{k}_i := [-y_i \; k_{1i} \cdots \; k_{mi}]', \quad \delta_{\hat{\alpha}} = [(\hat{\alpha}_0 - \hat{\alpha}_0^\iota) \;\; (\hat{\alpha}_1 - \hat{\alpha}_1^\iota) \cdots (\hat{\alpha}_m - \hat{\alpha}_m^\iota)]'. \tag{20}$$

The above equation, together with the equality constraint of (6), can be equivalently stated as a $(m + 1) \times (m + 1)$ matrix equation

$$A \delta_{\hat{\alpha}} = \mathbf{1} \delta_\lambda \tag{21}$$

where

$$A := \begin{pmatrix} 0 & -y_1 & \cdots & -y_m \\ -y_1 & k_{11} & \cdots & k_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ -y_m & k_{m1} & \cdots & k_{mm} \end{pmatrix}, \quad \mathbf{1} := \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \quad \text{and } \delta_\lambda = \lambda - \lambda^\iota. \tag{22}$$

Hastie's algorithm assumes that A is full rank and obtains

$$\delta_{\hat{\alpha}} = A^{-1} \mathbf{1} \delta_\lambda := d_p \delta_\lambda, \text{ or} \tag{23}$$

$$\hat{\alpha}_i(\delta_\lambda) = \hat{\alpha}_i^\iota + d_p \delta_\lambda \; \forall \; i \in \mathcal{E} \cup \{0\}. \tag{24}$$

Clearly, d_p in (23) corresponds to a *specific* direction and $d_p \delta_\lambda$ for $\delta_\lambda < 0$ corresponds to a one-dimensional movement of $\hat{\alpha}_i, i \in \mathcal{E}$ from $\hat{\alpha}_i^\iota$. Using (24), Hastie's algorithm determines the candidate values of $\lambda^{\iota+1}$ by evaluating the values of λ, when $\hat{\alpha}_i$ reaches 0 and 1 for every $i \in \mathcal{E}$. This is done by setting the left hand side of (24) to 0 and 1 respectively and solving for the corresponding values of δ_λ or λ. The algorithm also evaluates the values of λ when points in $\mathcal{L}$ approaches $\mathcal{E}$ due to the changes in $\hat{\alpha}_i, i \in \mathcal{E}$ by solving for the λ such that $\zeta_i = 0$ in (2) (a different notation and formula is used in their paper). The case of λ for points in $\mathcal{R}$ reaching $\mathcal{E}$ can be similarly determined using (2). Hence, with $|\mathcal{I}| = N$ and $|\mathcal{E}| = m$, a total of $2m + (N - m)$ candidate λ values are computed. The largest value of λ such that $\lambda < \lambda^\iota$ is set as $\lambda^{\iota+1}$. Using this procedure, Hastie's algorithm proceeds along until termination.

III. The Proposed Algorithm

For easy reference, the proposed algorithm is termed SVMP, short for SVM-Path. Its determination of $\lambda^{\iota+1}$ differs from Hastie's algorithm in that candidate values are not computed. Instead, the permissible range of λ is determined, including the case where the A matrix in (21) is rank deficient. More exactly, matrix A of (21) is represented by its Singular Value Decomposition,

$$A = U\Sigma V', \tag{25}$$

where $U := [u_1 \cdots u_m], u_i \in R^m, \Sigma =$diag$(\sigma_1, \cdots, \sigma_m)$ with $\sigma_i \geq \sigma_{i+1}$ for all $i = 1, \cdots, m-1$ and $V = U$ (as A is symmetric) and its rank r is determined by

$$r := max\{i : \sigma_i \geq \epsilon_{svd}\}. \tag{26}$$

for some $\epsilon_{svd} > 0$. Typically and as in SVMP, ϵ_{svd} is some measure of the underlying machine accuracy.

The general solution to (21) is, hence, given by

$$\delta_{\hat{\alpha}} = d_p \delta_\lambda + \sum_{r+1}^{m} u_j \beta_j := d_p \delta_\lambda + U_{m-r}\beta \tag{27}$$

where $d_p := V_r \Sigma_r^+ U_r' \mathbf{1}$ is the *unique* solution in the rowspace of A, $u_j, j = r+1, \cdots, m$ are the $m-r$ basis vectors of the nullspace of A and $\beta_j,\ j = r+1, \cdots, m$ are free variables, $U_r, U_{m-r}(V_r, V_{m-r}) \in R^{m \times r}$ are the matrices containing the first r and last $m-r$ columns of matrices U (V) respectively and $\Sigma_r^+ =$diag$(\sigma_1^{-1}, \cdots, \sigma_r^{-1})$ is the $r \times r$ diagonal matrix.

In addition to (27), constraints imposed by points in $\mathcal{L}$ and $\mathcal{R}$ have to be considered for the allowable movements of $\hat{\alpha}_j, j \in \mathcal{E}$ in between events. For this purpose, consider variable $\xi_i(\hat{\alpha}, \hat{\alpha}_0, \lambda)$ of (16) and optimality conditions (13)-(15). For a point i to stay in $\mathcal{R}$ due to changes in $\hat{\alpha}_j, j \in \mathcal{E}$ and λ, the condition $\xi_i(\hat{\alpha}, \hat{\alpha}_0, \lambda) \leq 0$ has to be maintained. Using this and subtracting two equations of (16) at λ and λ^ι yields

$$\delta_\lambda - \hat{k}_i' \delta_{\hat{\alpha}} + \xi_i^\iota \lambda^\iota \leq 0,\ \forall\ i \in \mathcal{R} \tag{28}$$

where $\hat{k}_i$ is that given by (20) and the notation $\xi_i^\iota := \xi_i(\hat{\alpha}^\iota, \hat{\alpha}_0^\iota, \lambda^\iota)$ is used in (28) and hereafter for simplicity. Using a similar development for $i \in \mathcal{L}$ leads to constraint

$$\delta_\lambda - \hat{k}_i' \delta_{\hat{\alpha}} + \xi_i^\iota \lambda^\iota \geq 0,\ \forall\ i \in \mathcal{L}. \tag{29}$$

Incorporating the allowable movement of $\delta_{\hat{\alpha}}$ given by (27), the determination of the next event can be posed as a Linear Programming (LP) problem over variables δ_λ and $\beta \in R^{m-r}$ parameterized by

$(\lambda^\iota, \hat{\alpha}^\iota, \xi^\iota)$ as

$$\min_{\delta_\lambda, \beta} \quad \delta_\lambda \tag{30}$$

$$\text{s.t. } 0 \leq \ \hat{\alpha}_i^\iota + d_p^i \delta_\lambda + \sum_{j=r+1}^{m} u_j^i \beta_j \leq 1 \ \forall \ i \in \mathcal{E} \tag{31}$$

$$p_i \delta_\lambda + \beta' q_i - \lambda^\iota \xi_i^\iota \geq 0 \ \forall \ i \in \mathcal{R} \tag{32}$$

$$p_i \delta_\lambda + \beta' q_i - \lambda^\iota \xi_i^\iota \leq 0 \ \forall \ i \in \mathcal{L} \tag{33}$$

$$\delta_\lambda \geq -\lambda^\iota \tag{34}$$

where d_p^i, u_j^i refer to the i^{th} element of d_p and u_j respectively,

$$p_i := d_p' \hat{k}_i - 1 \quad q_i := U_{m-r}' \hat{k}_i \tag{35}$$

and (34) is imposed to ensure that only $\lambda \geq 0$ is considered.

Suppose $(\delta_\lambda^*, \beta^*)$ is the minimizer of the LP. Then λ at the next event is defined by

$$\lambda^{\iota+1} := \delta_\lambda^* + \lambda^\iota. \tag{36}$$

The cross-over indices at $\lambda^{\iota+1}$ can be obtained by noting the active constraints in (31)-(34). These active constraints hold as equality when $(\delta_\lambda^*, \beta^*)$ are substituted into (31)-(34) and are used to update $\mathcal{E}$, $\mathcal{L}$ and $\mathcal{R}$ accordingly. To be precise, let $\mathcal{J}$ contains the indices of constraints that hold as equality in (31)-(34). The variables $\hat{\alpha}$ and ξ are updated according to

$$\tilde{\alpha}_i^{\iota+1} := \hat{\alpha}_i^\iota + d_p^i \delta_\lambda^* + \sum_{j=r+1}^{m} u_j^i \beta_j^* \quad \forall \ i \in \{0\} \cup \mathcal{E}. \tag{37}$$

$$\hat{\alpha}_i^{\iota+1} := \begin{cases} 0, & \tilde{\alpha}_i^{\iota+1} < 0; \\ 1, & \tilde{\alpha}_i^{\iota+1} > 1; \\ \tilde{\alpha}_i^{\iota+1}, & \text{otherwise.} \end{cases} \tag{38}$$

$$\xi_i^{\iota+1} = \frac{-p_i \delta_\lambda^* - (\beta^*)' q_i + \lambda^\iota \xi_i^\iota}{\lambda^{\iota+1}} \quad \forall \ i \in \mathcal{L} \cup \mathcal{R} \cup \mathcal{E} \tag{39}$$

where the clipping operation of (38) is needed due to numerical inaccuracies. The index set $\mathcal{J}$ is further

divided into

$$\mathcal{J}^{\mathcal{E}\to\mathcal{L}} = \{i : i \in \mathcal{J} \cap \mathcal{E}, \hat{\alpha}_i^{\iota+1} = 1\}, \tag{40}$$

$$\mathcal{J}^{\mathcal{E}\to\mathcal{R}} = \{i : i \in \mathcal{J} \cap \mathcal{E}, \hat{\alpha}_i^{\iota+1} = 0\}, \tag{41}$$

$$\mathcal{J}^{\mathcal{L}\to\mathcal{E}} = \{i : i \in \mathcal{J} \cap \mathcal{L}, \xi_i^{\iota+1} = 0\}, \tag{42}$$

$$\mathcal{J}^{\mathcal{R}\to\mathcal{E}} = \{i : i \in \mathcal{J} \cap \mathcal{R}, \xi_i^{\iota+1} = 0\} \tag{43}$$

and $\mathcal{E}, \mathcal{L}$ and R are updated as

$$\mathcal{E}^{\iota+1} = \mathcal{E}^{\iota} \cup \mathcal{J}^{\mathcal{L}\to\mathcal{E}} \cup \mathcal{J}^{\mathcal{R}\to\mathcal{E}} \backslash \mathcal{J}^{\mathcal{E}\to\mathcal{L}} \backslash \mathcal{J}^{\mathcal{E}\to\mathcal{R}}, \tag{44}$$

$$\mathcal{L}^{\iota+1} = \mathcal{L}^{\iota} \cup \mathcal{J}^{\mathcal{E}\to\mathcal{L}} \backslash \mathcal{J}^{\mathcal{L}\to\mathcal{E}}, \tag{45}$$

$$\mathcal{R}^{\iota+1} = \mathcal{R}^{\iota} \cup \mathcal{J}^{\mathcal{E}\to\mathcal{R}} \backslash \mathcal{J}^{\mathcal{R}\to\mathcal{E}}. \tag{46}$$

Remark 1 Clearly, β does not exist when A is full rank and the LP has to be modified in the most obvious way ($q_i = 0$ for $i \in \mathcal{L} \cup \mathcal{R}$ and $u_j^i = 0$ for $i \in \{0\} \cup \mathcal{E}$ in (31)-(34) and (37) - (39)) to be defined for the single variable δ_λ. In that case, the LP has only a single variable δ_λ and its solution can be expressed explicitly as

$$\delta_\lambda^* := \max\{\delta_{\lambda_{min}}^{\mathcal{E}}, \delta_{\lambda_{min}}^{\mathcal{L}}, \delta_{\lambda_{min}}^{\mathcal{R}}, -\lambda^{\iota}\} \text{ where} \tag{47}$$

$$\delta_{\lambda_{min}}^{\mathcal{E}} := \max\{\{\frac{1-\hat{\alpha}_i}{d_p^i} : d_p^i < -\tau, i \in \mathcal{E}\}, \{\frac{-\hat{\alpha}_i}{d_p^i} : d_p^i > \tau, i \in \mathcal{E}\}\}, \tag{48}$$

$$\delta_{\lambda_{min}}^{\mathcal{R}} := \max\{\frac{\lambda^{\iota}\xi_i^{\iota}}{p_i} : p_i > \tau, i \in \mathcal{R}\}, \quad \delta_{\lambda_{min}}^{\mathcal{L}} := \max\{\frac{\lambda^{\iota}\xi_i^{\iota}}{p_i} : p_i < -\tau, i \in \mathcal{L}\} \tag{49}$$

where $\tau > 0$ is a small quantity to prevent numerical overflow situation. It is worth noting that these conditions are different from those given in Hastie's approach.

When $m > r$, there are many solutions to DP, even when the minimizer $(\delta_\lambda^*, \beta^*)$ is unique. These possible solutions, for any choice of λ, $\lambda^{\iota+1} < \lambda \leq \lambda^{\iota}$, can be found from (31)-(34). Define

$$\Pi^{\iota} := \{(\delta_\lambda, \beta) : (\delta_\lambda, \beta) \text{ satisfies } (31) - (34) \text{ with parameters } \lambda^{\iota}, \hat{\alpha}^{\iota}, \xi^{\iota}\}, \tag{50}$$

$$\pi^{\iota}(\lambda) := \{\beta : \delta_\lambda = \lambda - \lambda^{\iota}, (\delta_\lambda, \beta) \in \Pi^{\iota}\}, \tag{51}$$

$$\hat{\alpha}_i^{\iota}(\lambda, \beta) := \hat{\alpha}_i^{\iota} + d_p^i(\lambda - \lambda^{\iota}) + \sum_{j=r+1}^{m} u_j^i \beta_j \ \forall \ i \in \mathcal{E} \cup \{0\}. \tag{52}$$

Theorem Suppose $\alpha_i^{\iota}, i \in \mathcal{E}^{\iota} \cup \mathcal{L}^{\iota} \cup \mathcal{R}^{\iota}$ satisfies the KKT conditions (12)-(15) at λ^{ι}. Let λ be given such that $\lambda^{\iota+1} < \lambda \leq \lambda^{\iota}$. Then the following results hold. (i) If $m > r$, $\hat{\alpha}_i^{\iota}(\lambda, \beta), i \in \{0\} \cup \mathcal{E}$ as given by (52) with any $\beta \in \pi^{\iota}(\lambda)$, together with $\hat{\alpha}_i = 0, i \in \mathcal{R}^{\iota}$ and $\hat{\alpha}_i = 1, i \in \mathcal{L}^{\iota}$ solves the Dual Problem

at λ. (ii) If $m > r$, then $(0,0) \in \Pi^\iota$. (iii) Suppose $\iota = 0$ and let $\lambda^{\iota+1}$, $\hat{\alpha}^{\iota+1}, \xi^{\iota+1}$, $\mathcal{E}^{\iota+1}, \mathcal{L}^{\iota+1}, \mathcal{R}^{\iota+1}$ be updated according to (36)- (46) ($q_i = 0$ for $i \in \mathcal{L}^\iota \cup \mathcal{R}^\iota$ and $u_j^i = 0$ for $i \in \{0\} \cup \mathcal{E}^\iota$ when $m = r$). Then $\{\lambda^\iota\} \rightarrow \lambda^\infty$ where $\{\lambda^\iota\}$ is the sequence generated from repeating the LP and update procedures. □

Proof. (i) Let $\delta_\lambda = \lambda - \lambda^\iota$ and $\hat{\alpha}_j = \hat{\alpha}_j^\iota(\lambda, \beta)$. Since $\beta \in \pi^\iota(\lambda)$, (δ_λ, β) satisfies the constraints (31)-(34). This, together with the condition that $\hat{\alpha}^\iota$ satisfies the KKT condition at λ^ι, means that (δ_λ, β) satisfies (21) or $\hat{\alpha}_i, i \in \{0\} \cup \mathcal{E}$ satisfy (18). This implies condition (12) of the KKT conditions is satisfied from (31). The remaining KKT conditions of (13) - (15) are satisfied as

$$\sum_{j\in\mathcal{L}}(\hat{\alpha}_j - \hat{\alpha}_j^\iota)k_{ji} + \sum_{j\in\mathcal{E}}(\hat{\alpha}_j - \hat{\alpha}_j^\iota)k_{ji} + \sum_{j\in\mathcal{R}}(\hat{\alpha}_j - \hat{\alpha}_j^\iota)k_{ji} - (\hat{\alpha}_0 - \hat{\alpha}_0^\iota)y_i \;\; (\geq) \;\; = \;\; [\leq] \; \lambda - \lambda^\iota \;\; \forall \; i \in (\mathcal{R}) \; \mathcal{E} \; [\mathcal{L}]$$

which, when added to (14) at λ^ι, shows the needed results. (ii) The stated result follows from (i) by choosing $\lambda = \lambda^\iota$. (iii) Since (0,0) is a feasible point of the LP from (ii), it follows that $\lambda^{\iota+1} \leq \lambda^\iota$ and this, together with $\lambda \geq 0$ implies that $\{\lambda^\iota\} \rightarrow \lambda_\infty$. □

Several observations are clear from the above theorem. The first is that feasibility of the LP holds at every ι from results (ii). The other is that choices of $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ are not unique for a specific value of λ when $m > r$. This follows from the non-uniqueness of $\delta_{\hat{\alpha}}$ in (27). For example, let $\partial\pi^\iota(\lambda)$ be the boundary set of $\pi^\iota(\lambda)$ and $\bar{\beta} \in \partial\pi^\iota(\lambda)$. Then there exists at least one i, say $\bar{i}$, with $\alpha_{\bar{i}}(\lambda, \bar{\beta}) = 0$ or 1. Correspondingly, $\bar{i}$ moves from $\mathcal{E}$ to $\mathcal{L} \cup \mathcal{R}$ as β moves from the interior of $\pi^\iota(\lambda)$ towards $\bar{\beta}$. For the case depicted Figure 1, $\partial\pi^\iota(\lambda)$ has two distinct elements resulting in 3 possible combinations of $\mathcal{E}, \mathcal{L}$, $\mathcal{R}$, one each when β takes on the value of an element of $\partial\pi(\lambda)$ and the third when $\beta \in int(\pi(\lambda))$. In the case where $\partial\pi^\iota(\lambda)$ is of dimension 1 or higher, many more combinations of $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ exist. Figure 1 shows the sets $\pi(\lambda)$ and $\partial\pi^\iota(\lambda)$ on the plot of δ_λ versus β for the case where $r = m - 1$.

Remark 2 Result (iii) of the above theorem shows that the sequence $\{\lambda^\iota\}$ converges to λ^∞. Clearly, this is not enough to guarantee that $\lambda^\infty = 0$. It is possible that the sequence converge to a λ^∞ where $\lambda^\infty > 0$. This can happen because of the non-uniqueness of $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ sets and the algorithm proceeds by switching between the various combinations of $\mathcal{E}, \mathcal{L}$ and R with no decrease in the value of λ. However, suppose $\lambda^\infty > 0$, then it follows from the existence of solution of DP at $\lambda = \lambda^\infty - \epsilon$ for any $\epsilon > 0$ that there exists at least one choice of $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ at λ^∞ for which descent on λ is possible. This provision is incorporated in SVMP even though the case of $\lambda^\infty > 0$ never happens in our experiment, suggesting the existence of many possible choices of $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ for which descent on λ is possible.

Remark 3 It follows from *Remark 2* that $\lambda^\infty = 0$ if $m = r$ for all events.

Remark 4 It is possible that $|\mathcal{E}| = 0$ after the update of $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ at a particular ι. Such a case is handled using a separate re-initialization routine in Hastie's approach. SVMP continues to proceed

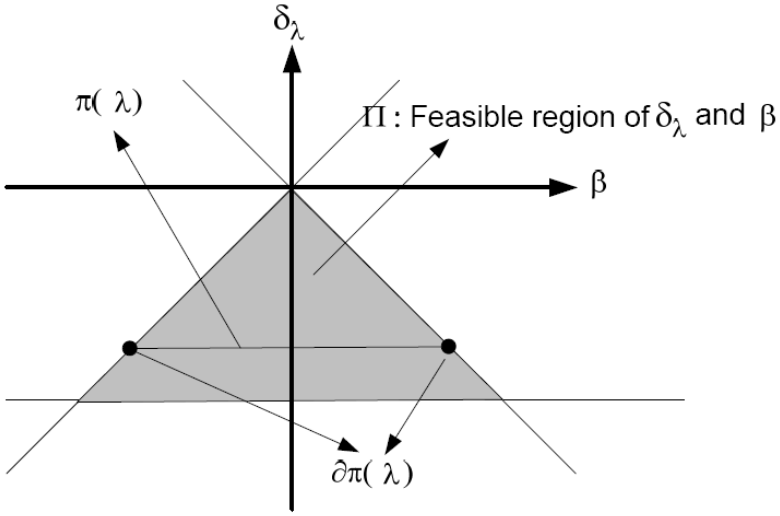

Fig. 1. The feasible region of δ_λ, β of LP, $\pi(\delta_\lambda)$ and $\partial\pi(\delta_\lambda)$

via the LP. Since there is no point in $\mathcal{E}$, (31) is non-existent. Let $d_p = 0$, $U_{m-r} = 0$ and this leads to $p_i = -1, q_i = 0\ \forall i \in \mathcal{L} \cup \mathcal{R}$ from (35). Correspondingly, the LP becomes $\min\{\delta_\lambda : \ -\delta_\lambda \geq \lambda^\iota \xi_i^\iota\ \forall\ i \in \mathcal{R},\ -\delta_\lambda \leq \lambda^\iota \xi_i^\iota\ \forall\ i \in \mathcal{L},\ \delta_\lambda \geq -\lambda^\iota\}$. As $\lambda^\iota \xi_i^\iota < 0$ for $i \in \mathcal{R}$, the corresponding constraints are redundant, leading to the solution $\delta_\lambda^* = \max\{\{-\lambda^\iota \xi_i^\iota : i \in \mathcal{L}\}, -\lambda^\iota\}$.

While (52) characterizes the full range of $\hat{\alpha}_i$ in terms of λ and β, a more convenient choice is to represent $\hat{\alpha}$ in terms of $\hat{\alpha}^\iota$ and $\hat{\alpha}^{\iota+1}$ for λ between λ^ι and $\lambda^{\iota+1}$. Such a parametrization corresponds to a particular choice of $\beta \in \pi(\delta_\lambda)$ and takes the form of

$$\hat{\alpha}_i(\lambda) := (1-\mu)\hat{\alpha}_i^\iota + \mu\hat{\alpha}_i^{\iota+1}, \mu = \frac{\lambda - \lambda^\iota}{\lambda^{\iota+1} - \lambda^\iota}\ \ \forall\ i \in \mathcal{I} \cup \{0\} \text{ and } \lambda^{\iota+1} < \lambda \leq \lambda^\iota \tag{53}$$

IV. Initialization and Termination

SVMP starts from an initial choice of $(\hat{\alpha}, \lambda^0)$ with $\lambda^0 = \bar{\lambda}$ where $\bar{\lambda}$ is any user-defined starting value for the regularization parameter and $\hat{\alpha}$ is the corresponding optimal minimizer of DP at $C = \bar{\lambda}^{-1}$. The value of $\hat{\alpha}$ is obtained using an Initialization Routine (IR). The Initialization Routine (IR) in SVMP is the well-known Sequential Minimal Optimization (SMO) of [3] incorporating the modifications of [4] and [13], although it can also be any procedure that solves DP at $\bar{\lambda}$. The SMO algorithm is used because it allows initialization at any $\bar{\lambda}$ and works for data set having duplicate or linearly dependent

points. As a review, SMO assumes the availability of $\{\alpha_i, i \in \mathcal{I} \cup \{0\}\}$ satisfying $\sum_{i\in\mathcal{I}} \alpha_i y_i = 0$ and proceeds by selecting two violating indices from two index sets $\mathcal{E} \cup \mathcal{R}^+ \cup \mathcal{L}^-$ and $\mathcal{E} \cup \mathcal{R}^- \cup \mathcal{L}^+$ where $\mathcal{R}^+ := \{i : i \in \mathcal{R},\ y_i = +1\}$, $\mathcal{R}^- := \{i : i \in \mathcal{R},\ y_i = -1\}$, $\mathcal{L}^+ := \{i : i \in \mathcal{L},\ y_i = +1\}$, $\mathcal{L}^- := \{i : i \in \mathcal{L},\ y_i = -1\}$. These two indices are $i_{up} := arg\max_i\{-f_i : i \in \mathcal{E} \cup \mathcal{R}^+ \cup \mathcal{L}^-\}$ and $i_{low} := arg\min_i\{-(f_i - f_{i_{up}})^2/\bar{a}_{i_{up}i} : i \in \mathcal{E} \cup \mathcal{R}^- \cup \mathcal{L}^+ \ , f_i > f_{i_{up}}\}$, where $f_i := \sum_{j\in\mathcal{I}} \alpha_j y_j k_{ji} - y_i$, $\bar{a}_{i_{up}i} := a_{i_{up}i}$ if $a_{i_{up}i} > 0$ and $\bar{a}_{i_{up}i} := 10^{-12}$ otherwise, and $a_{i_{up}i} := k_{ii} + k_{i_{up}i_{up}} - 2k_{i_{up}i}$. Using them at every iteration until termination, SMO does a joint optimization of $(\alpha_{i_{up}}, \alpha_{i_{low}})$ to reduce the objective function (4) while maintaining the equality constraint and updates all necessary $f_i, i \in \mathcal{I}$. This process is repeated recursively until the satisfaction of the KKT condition which can be equivalently stated as

$$f_j - f_{i_{up}} \leq \epsilon_{smo} \tag{54}$$

for some $\epsilon_{smo} > 0$, where $f_j := \min_i\{-f_i : i \in \mathcal{E}\cup\mathcal{R}^-\cup\mathcal{L}^+\}$. The exact equations needed to implement the SMO algorithm can be found in [3], [4] and [13].

The input variables to IR are $\hat{\alpha}_i^0, i \in \mathcal{I}$ and λ^0 where λ^0 is some very large positive number and $\hat{\alpha}^0$ is an estimate of the optimal $\hat{\alpha}$. The outputs are the optimal values of $\hat{\alpha}_i, i \in \mathcal{I}$, and their corresponding sets $\mathcal{L}, \mathcal{R}, \mathcal{E}$ at λ^0. Let $\mathcal{I}^+ := \{i : y_i = +1\}$, $\mathcal{I}^- := \{i : y_i = -1\}$. Assume that $N_d := |\mathcal{I}^+| - |\mathcal{I}^-| > 0$ and define $\mathcal{I}^+_{N_d}$ be the index set containing the first N_d indices of $\mathcal{I}^+$. The values of $\hat{\alpha}_i^0, i \in \mathcal{I}$ as input to IR are chosen as

$$\hat{\alpha}_i = 1\ \forall i \in \mathcal{I}^+\backslash\mathcal{I}^+_{N_d} \cup \mathcal{I}^-,\ \hat{\alpha}_i = 0\ \forall i \in \mathcal{I}^+_{N_d} \tag{55}$$

which ensures the satisfaction of the equality constraint in (12). It is important to note that the computations needed by IR is modest because very large value of λ^0 allows easy satisfaction of constraint (2). The SMO algorithm is also used as a Backup Routine (BR) and it is invoked using a different set of initial values. The motivation and the details of its use are discussed in section V.

Similarly, SVMP accepts any user-defined $\underline{\lambda}$ as the lowest value of λ for SVMP. Hence, there is only one stopping condition in SVMP: $\lambda^\iota < \underline{\lambda}$. As observed in [9], there is no more new event when $|\mathcal{L}| = 0$. This condition can be easily verified to be true from (12)-(15). Suppose $|\mathcal{L}^\iota| = 0$ for some ι, then $\lambda^{\iota+1} = \underline{\lambda}$ with $\hat{\alpha}_i^{\iota+1} = \underline{\lambda}\hat{\alpha}_i^\iota/\lambda^\iota$ for $i \in \mathcal{E}^\iota$ and $\hat{\alpha}_i^{\iota+1} = 0$ for $i \in \mathcal{R}^\iota$. Also, a default value of $\underline{\lambda} = \tau$, where $\tau > 0$ is a small tolerance, is used in the absence of a user input.

It is now possible to give the outline of the Basic Algorithm (BA) of SVMP:

Given $\bar{\lambda}$, $\underline{\lambda}$, ϵ_{svd}

1. Initialization :

Call IR with $\hat{\alpha}$ according to (55) and $\lambda = \bar{\lambda}$.

2. Main loop:

While $\lambda > \underline{\lambda}$,

a. If $|\mathcal{E}| = 0$,

set $d_p = 0, U_{m-r} = 0, p_i = -1, q_i = 0$ for all $i \in \mathcal{L} \cup \mathcal{R}$

else

obtain $A = U\Sigma V'$ using (22) and determine r from (26) with ϵ_{svd},

compute d_p, U_{m-r} using (27), $\hat{k}_i \ \forall \ i \in \mathcal{I}$ using (20) and $p_i, q_i \ \forall \ i \in \mathcal{L} \cup \mathcal{R} \cup \mathcal{E}$ using (35).

end

b. Solve LP of (30)-(34).

c. Update $\mathcal{E}, \mathcal{L}, \mathcal{R}, \alpha_i, \lambda, \xi_i$ according to (36) - (46).

end

V. Implementation Issues and The Backup Routine

BA works well for most data sets. For data sets having many duplicate points, it is possible that the behavior of "looping" as described by *Remark 2* may happen and provision for handling this situation is incorporated into SVMP. If looping occurs, a Backup Routine (BR) is used to identify the points in $\mathcal{E}, \mathcal{L}$ and R so that descent on λ can continue. Our choice of BR is also the SMO algorithm. When "looping" is detected at λ^ι, BR is invoked with $\hat{\alpha}_i^\iota$ but for $\lambda = (1-\gamma)\lambda^\iota$ for some small $\gamma > 0$. Since $\hat{\alpha}_i^\iota$ is optimal at λ^ι, the SMO algorithm converges quickly. Such "warm start" strategy for speeding up the algorithm is well known [14, 15]. Upon return, the SMO algorithm provides the optimal $\hat{\alpha}_i^\iota, i \in \{0\} \cup \mathcal{I}$, $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ at $(1-\gamma)\lambda^\iota$.

Since (12) - (15) can only be satisfied up to some tolerance in numerical computations, it is possible for small violations of these KKT conditions to happen during tracking in BA, especially when high accuracy of the solutions of DP are needed. These violations happen due to the accumulation of numerical errors, arising from the use of various tolerances and the incremental nature of updates of ξ and $\hat{\alpha}$ in (37)-(39). The use of tolerances are inevitable in numerical computations and it is not clear if much more can be done to improve accuracy. The other source of error is the incremental updates of ξ and $\hat{\alpha}$, which do not offer opportunity to correct initial or accumulated errors. One way to minimize such error is to compute ξ using $\hat{\alpha}^\iota$ and λ^ι in (16) instead of (39). This, of course, is more expensive. Another is to break a large range of λ into union of smaller ranges and running SVMP several times, each over a small range to avoid errors from building up. Of course, error also accumulates in $\hat{\alpha}$ and they have to be readjusted when detected. Such errors can be detected by verifying the satisfaction of KKT conditions after step c.

in BA. By caching ξ_i, the KKT verification can be done cheaply via the use of a positive tolerance, ϵ_{kkt}, in the form of

$$|\xi_i| \leq \epsilon_{kkt} \ \forall i \in \mathcal{E}, \quad \xi_i \leq \epsilon_{kkt} \ \forall i \in \mathcal{R}, \quad \xi_i \geq -\epsilon_{kkt} \ \forall i \in \mathcal{L}. \tag{56}$$

Upon violation of (56), the BR is invoked and $\hat{\alpha}, \xi$ are recomputed. In this case, $\hat{\alpha}$ obtained as the solution of LP at λ^ι is already available and is used as the input to BR. Again, due to the closeness of $\hat{\alpha}_i$ to the optimal, the convergence of SMO is fast. Our numerical experiment suggest that such violation of KKT can be minimized when $\hat{\alpha}$ at $\iota = 0$ is accurately determined. This can be achieved by using a tight tolerance ϵ_{smo} for IR (tighter than that for BR) to impose greater accuracy requirement from the start. Of course, this KKT check is not necessary if a less accurate solution is acceptable.

VI. Special-Event Routine

In most data sets, BR is rarely invoked. When invoked, it normally converges quickly except when λ is very small. It is therefore useful to minimize the number of times BR is invoked. SVMP does so by using a few heuristics, collected in a Special-Event Routine, SR. SR is invoked just before calling the LP solver at step b. and it is designed to minimize the use of BR for the "looping" situation. It looks for inequalities that blocks descent of δ_λ in the LP. Specifically, it checks for i that satisfy the following conditions:

- (S1) $d_p^i < -\tau$, $\hat{\alpha}_i^\iota > 1 - \tau$, $\dim(U_{m-r}) = 0$ and $(1 - \hat{\alpha}_i^\iota)/d_p^i > -\tau_2 \ \ \forall \ \ i \in \mathcal{E}$
- (S2) $d_p^i > \tau$, $\hat{\alpha}_i^\iota < \tau$, $\dim(U_{m-r}) = 0$ and $-\hat{\alpha}_i^\iota/d_p^i > -\tau_2 \ \ \forall \ \ i \in \mathcal{E}$
- (S3) $p_i > \tau$, $\dim(U_{m-r}) = 0$, $-\xi_i^\iota < \tau$ and $\lambda^\iota \xi_i^\iota / p_i > -\tau_2 \ \ \forall \ \ i \in \mathcal{R}$
- (S4) $p_i < -\tau$, $\dim(U_{m-r}) = 0$, $\xi_i^\iota < \tau$ and $\lambda^\iota \xi_i^\iota / p_i > -\tau_2 \ \ \forall \ \ i \in \mathcal{L}$

for some small tolerances τ and τ_2. It is easy to see that points satisfying any one condition of (S1) - (S4) will block LP from making progress to reduce δ_λ. These points are also those with $\hat{\alpha}_i \approx 0$ or 1 for $i \in \mathcal{E}$, $\xi_i \approx 0$ for $i \in \mathcal{L}$ and $\xi_i \approx 0$ for $i \in \mathcal{R}$ which are susceptible to wrong assignment due to numerical errors. Hence, a point such that (S1) or (S2) is satisfied is moved from $\mathcal{E}$ to $\mathcal{L}$ and $\mathcal{R}$ respectively while a point satisfying (S3) or (S4) is moved from $\mathcal{R}$ or $\mathcal{L}$ to $\mathcal{E}$ respectively. It is important to note that such a reassignment does not violate the satisfaction of the KKT conditions since a point with $\hat{\alpha}_i = 0, \xi_i = 0 (\hat{\alpha}_i = 1, \xi_i = 0)$ can be classified as either in $\mathcal{E}$ or $\mathcal{R}(\mathcal{L})$. When this happens the LP

is not called. Instead, SVMP updates the sets of $\mathcal{E}, \mathcal{L}$ and $\mathcal{R}$ according to

$$\mathcal{S}^{\mathcal{E}\to\mathcal{L}} := \{i : i \text{ satisfies (S1)}\}, \tag{57}$$

$$\mathcal{S}^{\mathcal{E}\to\mathcal{R}} := \{i : i \text{ satisfies (S2)}\}, \tag{58}$$

$$\mathcal{S}^{\mathcal{R}\to\mathcal{E}} := \{i : i \text{ satisfies (S3)}\}, \tag{59}$$

$$\mathcal{S}^{\mathcal{L}\to\mathcal{E}} := \{i : i \text{ satisfies (S4)}\}, \tag{60}$$

$$\mathcal{E}^{\iota+1} := \mathcal{E}^{\iota} \cup \mathcal{S}^{\mathcal{L}\to\mathcal{E}} \cup \mathcal{S}^{\mathcal{R}\to\mathcal{E}} \backslash \mathcal{S}^{\mathcal{E}\to\mathcal{L}} \backslash \mathcal{S}^{\mathcal{E}\to\mathcal{R}}, \tag{61}$$

$$\mathcal{L}^{\iota+1} := \mathcal{L}^{\iota} \cup \mathcal{S}^{\mathcal{E}\to\mathcal{L}} \backslash \mathcal{S}^{\mathcal{L}\to\mathcal{E}}, \tag{62}$$

$$\mathcal{R}^{\iota+1} := \mathcal{R}^{\iota} \cup \mathcal{S}^{\mathcal{E}\to\mathcal{R}} \backslash \mathcal{S}^{\mathcal{R}\to\mathcal{E}} \tag{63}$$

A simple mechanism is also incorporated to ensure that infinite switching of points between $\mathcal{E}$ and $\mathcal{L} \cup \mathcal{R}$ is avoided. The SR has the advantage of minimizing "looping", reducing the number of events and hence minimizing the number of times BR is called due to numerical errors.

The outline of the SVMP algorithm is stated below.

Given $\bar{\lambda}$, $\underline{\lambda}$, $\epsilon_{svd}, \epsilon_{kkt}, \epsilon_{smo}, \tau, \tau_2, \gamma, \ell$

1. Initialization :

Set $\iota = 0$, $n_{blk} = 0$, $n_{SR} = 0$, $flag_{SR} = 0$ and $\lambda_c = \lambda^0 = \bar{\lambda}$.

Call IR with λ^0 and $\hat{\alpha}$ according to (55) with ϵ_{smo}.

2. Main:

While $\lambda > \underline{\lambda}$,

If $|\mathcal{E}| = 0$,

set $d_p = 0, U_{m-r} = 0, p_i = -1, q_i = 0$ for all $i \in \mathcal{L} \cup \mathcal{R}$

else

obtain $A = U\Sigma V'$ using (21) and determine r from (26) with ϵ_{svd},

compute d_p, U_{m-r} using (27), $\hat{k}_i \ \forall \ i \in \mathcal{I}$ using (20) and $p_i, q_i \ \forall \ i \in \mathcal{L} \cup \mathcal{R} \cup \mathcal{E}$ using (35).

endif

If $n_{SR} \leq \ell$

invoke SR on (31)-(34) according to (S1) -(S4) using τ and τ_2.

If none of (S1) -(S4) is satisfied,

set $flag_{SR} = 0$ and $n_{SR} = 0$

else

set $n_{SR} = n_{SR} + 1$, $flag_{SR} = 1$

Update $\mathcal{E}, \mathcal{L}, \mathcal{R}$ according to (57) - (63).

endif

endif

If $flag_{SR} = 0$,

solve LP of (30)-(34) using (47)-(49) when $m = r$ or a LP solver when $m > r$.

If $n_{blk} > \ell$

invoke BR with $\hat{\alpha}_i, i \in \mathcal{I}$ at $\lambda = (1 - \gamma)\lambda^{\iota}$.

Update $\mathcal{E}, \mathcal{L}, \mathcal{R}, \alpha, \lambda, \xi$ based on the results of BR.

set $\lambda_c = \lambda$, $n_{blk} = 0$

else

If $\lambda_c - \lambda < \tau_2$

$n_{blk} = n_{blk} + 1$

else

$n_{blk} = 0, \lambda_c = \lambda$

Update $\mathcal{E}, \mathcal{L}, \mathcal{R}, \alpha, \lambda, \xi$ according to (36) - (46).

endif

`OPTIONAL` Check KKT conditions of (56) using ϵ_{kkt}. If KKT condition is violated, call BR with $\hat{\alpha}$ and λ.

endif

endif

$\iota = \iota + 1$.

end

VII. COMPUTATIONAL SPEEDUP

The most computationally expensive step in SVMP is the Singular Value Decomposition of A of (25), especially when m is large. The SVD decomposition gives the most accurate determination of the rank of A but it is also the most expensive. Depending on its exact implementation, SVD decomposition has an approximate computational complexity of $O(12m_A^2)$ where m_A is size of A. An alternative to SVD is the well-known QR decomposition with column pivoting [16] where the complexity is about $O(\frac{4}{3}m_A^2)$ [17]. Here, the columns of A are pivoted with permutation matrix E so that

$$AE = QR \tag{64}$$

where Q is an orthogonal matrix and R is an upper triangular matrix with diagonal elements in decreasing value ($R_{i,i} \geq R_{i+1,i+1}$ for all i). Hence, the rank of A can be determined from the norm of the rows of R using an appropriate tolerance. Suppose rank of A is r, it follows from (64) that the basis vectors of the nullspace of A are given by the last $m-r$ columns of Q. While different from those obtained from the SVD decomposition, these vectors span the same null space. The value of d_p of (27) is obtained via the standard three-stage approach: by first solving for g of $Qg = \mathbf{1}$, followed by h of $Rh = g$ and $d_p = Eh$. The first stage is solved noting that $Q' = Q^{-1}$ while the second is by backward substitution, exploiting the upper triangular structure of R. The last stage of $d_p = Eh$ requires only a reassignment of the vector of h to d_p and is not a matrix-vector multiplication. This d_p is not necessarily the same as that obtained from the SVD of A as it contains components from both the rowspace and nullspace of A. Obviously, the LP solution is not affected by these different values of d_p and U_{m-r} and the proposed algorithm proceeds as in the case with SVD.

It is well known that when A is changed only slightly (either by an addition/deletion of a row/column or an update), insertion, deletion and update formula can be used to compute d_p without recomputing the QR decomposition from scratch. Such formulae have $O(m_A^2)$ complexity and their applications are very well documented [17, 18], hence, their discussion is omitted here. Instead, considerations of the applicability of these formulae under the current context are discussed. Standard QR update formulae for addition/deletion of column or row *do not* preserve the decreasing order of the diagonal elements of R. Consequently, updated QR decomposition of A should not be used for determination of its rank. These formula should only be applied in the restricted situation where it is known a priori that A is non-singular and rank determination is not needed. For example, such a situation corresponds to the case when A is full rank at a prior event and the solution of the LP corresponds to the case where a point moves from $\mathcal{E}$ to $\mathcal{L} \cup \mathcal{R}$. In this case, the new $\mathcal{E}$ has one less element than the previous and the corresponding A

matrix would be non-singular. The solution of d_p can then be obtained by removing a row and a column successively.

The insertion formula to update QR when a point is added to $\mathcal{E}$ can also be used but this should be done with care since the subsequent determination of the rank of A is not accurate. The idea is to use an indicator which forewarns rank deficiency should the insertion proceeds. When such a situation is detected, insertion formula should be avoided and QR decomposition (with column pivoting) of A is done from scratch. The choice of the indicator function used is $s := (1 + v'A^{-1}u)$ where v, u are the column vectors of appropriate dimension in the updated matrix, $(A + uv')$, whose QR decomposition is to be computed. When s is close to zero, QR decomposition from scratch is done. The performance enhancement due to the use of updating formula are shown in the next section.

VIII. Experiments

The experiment is done on an AMD Athlon (tm) 64 $\times$ 2 Duo Core processor 5000+ running at 2.61 GHz with 4GB of memory under the Windows XP Operating System. The environment in which these experiment were conducted is Matlab 2007. Since implemented in Matlab, the computational time is expected to be slower than if the algorithm is implemented in C or other languages. The intention is not to compare with other algorithms on speed but to show the applicability of the algorithm to all data sets, on data sets that are larger in size and the relative saving in time using the different solvers of the A matrix. A copy of the Matlab code is available at http://guppy.mpe.nus.edu.sg/ mpeongcj/ongcj.html. Data sets and their characteristics used in our experiments are given in Table I and can be obtained from UCI repository [19]. Each feature vector is normalized to be zero mean and unit variance for all experiments. Two choices of kernel function are used: Gaussian where $k_{i,j} = \frac{||x_i - x_j||}{\sigma}$ and linear where $k_{i,j} = x_i' x_j$. Unless otherwise specified, the rest of the parameters and settings used in the experiments are: $\ell = 50$, $\gamma = 0.01$, $\bar{\lambda} = 10^4$, $\underline{\lambda} = 10^{-3}$, $\epsilon_{svd} = m * eps$ where $eps = 10^{-14}$ is an estimate of the machine accuracy, $\epsilon_{kkt} = 10^{-3}$ for BR, $\epsilon_{kkt} = 10^{-6}$ for IR, $\tau = 10^{-4}, \tau_2 = 10^{-8}$, kernel function is linear, $\sigma = 0.5$ if Gaussian kernel is used, decomposition of A is SVD, ξ_i is updated based on (39) and the LP solver is an active set based variant of the sequential quadratic programming algorithm based on the strategy by [20] and is available in the Matlab environment.

Data set	N	n	Data set	N	n
Sonar	104	60	Diabetes	468	8
Heart	170	13	HillValley	606	100
Ionosphere	200	33	German	800	27
Wisconsin Breast Cancer (wbc)	350	9	svmguide3	1243	21
Monk 1	432	6	madelon	2000	500
Monk 2	432	6	splice-c†	3175	60
Monk 3	432	6	svmguide-c†	7089	4

Table I. The number of points and features of data sets used in the experiment. † splice-c and svmguide-c are sets containing both training and testing points of splice and svmguide data sets respectively.

Tables II and III show the results of SVMP on data sets with $N < 1000$ for the linear and Gaussian kernel respectively. The quantities $|\mathcal{E}|_{max}, (m-r)_{max}, \iota_{max}$ refer to the maximal $|\mathcal{E}|$, $dim(U_{m-r})$ over all events and the total number of events between $\bar{\lambda}$ and $\underline{\lambda}$ respectively. The quantities t_{IR}, t_S^{SVD}, t_S^{QR}, t_S^{UD} refer to the CPU times used for the Initialization Routine, the solutions of the SVMP for $\lambda = 10^4$ to $\lambda = 10^{-3}$ (excluding the time for IR and the time for computing the kernel matrix) using SVD, QR and the QR update formula for the solution of the A matrix respectively. N_{SR} and N_{kkt} refer to the number of times Special-Event Routine and Backup Routine (due to a violation of KKT condition) are invoked respectively. These two tables also include quantities to show the accuracy of the solution obtained from SVMP relative to that obtained using LIBSVM. This is done by evaluating the optimal cost, $c = \frac{1}{2}w(\alpha)'w(\alpha) + \lambda^{-1}\sum\zeta_i$, obtained from these two methods at 100 random values of λ and showing the maximal relative difference over these values, indicated by $c_{max}^{SL} := |\frac{c_S - c_L}{c_L}|_{max}$ with c_S and c_L being the costs obtained using SVMP and LIBSVM respectively. The quantity c_{max}^{SL2} in Table III is defined similarly as c_{max}^{SL} except that it refers to the case where the optional KKT check is turned off.

Data set	$\|\mathcal{E}\|_{max}$	ι_{max}	$(m-r)_{max}$	t_{IR}, $t_S^{SVD}, t_S^{QR}, t_S^{UD}$ (sec)	$c_{max}^{SL}\%$	N_{SR}	N_{kkt}
sonar	42	207	0	0.150, 1.046, 1.030, 1.290	0.2153	1	0
heart	14	336	0	0.156, 1.593, 1.625, 1.891	0.003	1	0
Ionosphere	34	388	0	0.234, 1.968, 1.935, 2.437	0.0233	1	0
wbc	11	507	6	0.250, 4.328, 4.297, 4.515	0.0075	30	0
Monk 1	105	52	98	0.172, 5.703, 11.82, 12.29	0.0012	7	0
Monk 2	290	1	283	0.203, 2.687, 2.328, 2.359	0.0004	0	0
Monk 3	119	74	86	0.203, 5.765, 6.062, 6.000	0.0033	24	0
diabete	9	350	0	0.281, 3.578, 3.687, 3.906	0.0004	1	0
HillValley	17	610	0	0.218, 9.453, 9.312, 9.931	0.0003	2	0
German	28	440	3	1.046, 12.97, 13.18, 13.85	0.0003	0	4

Table II. Performance of SVMP on data sets with $N < 1000$ using linear kernel . The quantities t_S^{SVD}, t_S^{QR} and t_S^{UD} are for all solutions between $\bar{\lambda} = 10^4$ to $\underline{\lambda} = 10^{-3}$.

Data set	$\|\mathcal{E}\|_{max}$	ι_{max}	$(m-r)_{max}$	$t_{IR}, t_S^{SVD}, t_S^{QR}, t_S^{UD}$ (sec)	$c_{max}^{SL}\%$	$c_{max}^{SL2}\%$	N_{SR}	N_{kkt}
sonar	104	8	0	0.234, 0.265, 0.265, 0.312	0.0823	30.06	2	1
heart	170	55	0	0.312, 2.031, 1.218, 1.468	0.0832	11.00	6	1
Ionosphere	193	13	1	0.312, 4.015, 2.328, 2.312	0.2073	10.82	0	1
wbc	195	285	29	0.250, 28.17, 20.62, 20.73	0.0683	0.0657	3	0
Monk 1	432	32	0	0.171, 21.43, 3.312, 3.708	0.0089	0.0088	32	0
Monk 2	432	143	0	0.687, 173.1, 20.65, 13.42	0.1818	13.74	0	1
Monk 3	432	270	0	0.391, 211.1, 27.18, 20.62	0.0965	3.667	7	1
diabete	457	180	0	0.750, 214.5, 27.97, 18.10	0.0674	7.703	1	3
HillValley	172	641	2	0.265, 42.65, 23.65, 27.21	0.0016	0.0198	18	11
German	800	112	0	2.680, 728.2, 92.20, 41.48	0.2456	22.256	15	1

Table III. Performance of SVMP on data sets with $N < 1000$ using Gaussian kernel with $\sigma = 0.5$. The quantity t_S^{SVD}, t_S^{QR} and t_S^{UD} are for all solutions between $\bar{\lambda} = 10^4$ to $\underline{\lambda} = 10^{-3}$.

Several observations can be drawn from Tables II to III. The CPU time needed for SVMP with Gaussian kernel is general higher than that for linear kernel. This is consistent with the larger values of $|\mathcal{E}|$ and ι_{max} for data sets with Gaussian kernel. For data sets where $|\mathcal{E}|_{max}$ is small, the computational time is modest, even when N is in the region of a few thousands. This is consistent with the observation that the Singular Value Decomposition (SVD) of A becomes the dominant factor for computational cost. The use of QR decomposition (with column pivoting) of A brings the computational cost for the Gaussian kernel case down by 11-83%. The reduction is much less (more in some cases) for the case of linear kernel. The use of QR update formula results in a smaller gain from the basic QR. In fact, for linear kernel,

the use of update results in a slight gain due to the small $|\mathcal{E}|$ size for most events and the additional overhead needed to implement the update formula. For Gaussian kernel, the $|\mathcal{E}|_{max}$ are always larger than the corresponding linear case and the QR update brings an advantage ranging from -17 to 94% compared to the SVD timing with the greater advantage coming from data sets having larger values of $|\mathcal{E}|_{max}$ and ι_{max}.

The number of KKT violations in the two tables also suggests that KKT violation should be checked when Gaussian kernel is used. Otherwise the error can be quite substantial, as seen by the entries in the c_{max}^{SL2} column. The KKT violation check for data sets using linear kernel is less important, as very few violations occur.

The next table shows computational times of SVMP on some larger data sets using the linear and Gaussian kernels under the same conditions as that for Tables II and III. For the case of linear kernel, it is possible to replace IR and BR with some recently developed code specially for linear kernel [5, 6, 8] to speed up the performance. This, however, is not done to preserve the applicability of SVMP to general kernel functions.

kernel	linear				Gaussian			
Data set	$\|\mathcal{E}\|_{max}$	ι_{max}	$(m-r)_{max}$	t_{IR}, t_S^{UD} (sec)	$\|\mathcal{E}\|_{max}$	ι_{max}	$(m-r)_{max}$	t_{IR}, t_S^{UD}(sec)
svmguide3	22	920	0	1.109, 58.28	1223	266	0	4.796, 173.3
madelon	501	4061	0	0.468, 754.8	2000	2	0	0.453, 2.345
splice-c	62	6340	3	1.609, 674.1	3149	156	179	18.82, 6964
svmguide-c	6	13287	2	17.40, 1472	606	14107	2	15.65, 4160

Table IV. Performance of SVMP for larger data sets using linear and Gaussian kernel for the range of 10^4 to 10^{-3}.

IX. Conclusion

This paper describes an improved algorithm for the computation of the solution of SVM classification problem for the entire path of the regularization parameter. It improves upon Hastie's algorithm in dealing with data sets having duplicate points or linearly dependent points in the kernel space. Central to the algorithm is a standard LP formulation for all tracking steps. Additional considerations include routines to deal with the accumulation of numerical errors, heuristics and updating formula to speed up algorithm. When the computations are done with QR and QR update formula, the corresponding saving in time relative to the SVD computations can be up to 90%, especially when there are large number of events and large $\mathcal{E}$ set.

REFERENCES

[1] 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.

[2] T. Joachims, *Making large-Scale SVM Learning Practical.* MIT Press, 1998, ch. In B. Scholkopf, C. Burges and A. Smola (Eds), Advances in kernel methods: Support Vector Learning.

[3] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*. MIT Press, 1998, ch. In B. Scholkopf, C. Burges and A. Smola (Eds), Advances in kernel methods: Support Vector Learning.

[4] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[5] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett, "Exponentiated gradient algorithms for conditional random fields and max-margin markov networks," *Journal of Machine Learning Research*, vol. 9, pp. 1775–1822, 2008.

[6] Q. V. L. Alexander J. Smola, S.V. N. Vishwanathan, "Bundle methods for machine learning," in *Advances in neural information processing systems(NIPS)*, 2008.

[7] S. S. Keerthi and D. DeCoste, "A modified finite newton method for fast solution of large scale linear svms," *J. Mach. Learn. Res.*, vol. 6, pp. 341–361, 2005.

[8] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm," in *ICML '08: Proceedings of the 25th international conference on Machine learning*. New York, NY, USA: ACM, 2008, pp. 408–415.

[9] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," *Journal of Machine Learning Research*, vol. 5, pp. 1391 –1415, October 2004.

[10] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," *ANNALS OF STATISTICS*, vol. 35, p. 1012, 2007.

[11] L. Gunter and J. Zhu, "Efficient computation and model selection for the support vector regression," *Neural Computation*, vol. 19, pp. 1633–1655, 2007.

[12] B. Scholkopf and A. J. Smola, *Learning with Kernels*. Cambridge MIT Press, 2002.

[13] R. Fan, P. Chen, and C. Lin, "Working set selection using second order information for training support vector machines," *Journal of Machine Learning Research*, vol. 6, pp. 1889 – 1918, Nov 2005.

[14] D. Decoste and Wagstaff, "Alpha seeding for support vector machines," in *Proceedings of the sixth*

ACM SIGKDD international conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM, 2000, pp. 345–349.

[15] M. Lee, S. Keerthi, C. Ong, and D. DeCoste, "An efficient method for computing leave-one-out error in support vector machines with gaussian kernels," *IEEE Transactions on Neural Network*, vol. 15, no. 3, pp. 750 – 757, May 2004.

[16] P. Businger and G. Golub, "Linear least squares solutions by householder transformations," *Numer. Math.*, vol. 7, pp. 269 – 276, 1965.

[17] H. G. Golub and C. F. V. Loan, *Matrix Computations*. John Hopkins, 1989.

[18] L. Trefethen and D. Bau, *Numerical Linear Algebra*. Society of Industrial and Applied Mathematics, 1997.

[19] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[20] P. Gill, W. Murray, and M. Wright, *Numerical Linear Algebra and Optimization, Vol. 1*. Addison Wesley, 1991.