Suboptimal Solution Path Algorithm for Support Vector Machine

Masayuki Karasuyama KRSYM@GOAT.ICS.NITECH.AC.JP
Ichiro Takeuchi TAKEUCHI.ICHIRO@NITECH.AC.JP
Nagoya Institute of Technology, Gokiso-cho, Syowa-ku, Nagoya, Aichi, 466-8555, JAPAN

Abstract

We consider a *suboptimal solution path algorithm* for the Support Vector Machine. The solution path algorithm is an effective tool for solving a sequence of a parametrized optimization problems in machine learning. The path of the solutions provided by this algorithm are very accurate and they satisfy the optimality conditions more strictly than other SVM optimization algorithms. In many machine learning application, however, this strict optimality is often unnecessary, and it adversely affects the computational efficiency. Our algorithm can generate the path of suboptimal solutions within an arbitrary user-specified tolerance level. It allows us to control the trade-off between the accuracy of the solution and the computational cost. Moreover, We also show that our suboptimal solutions can be interpreted as the solution of a *perturbed optimization problem* from the original one. We provide some theoretical analyses of our algorithm based on this novel interpretation. The experimental results also demonstrate the effectiveness of our algorithm.

1. Introduction

Recently, the *solution path algorithm* (Efron et al., 2004; Hastie et al., 2004) has been widely recognized as one of the effective tools in machine learning. It can efficiently compute a sequence of the solutions of a parametrized optimization problem. This technique is originally developed as *parametric programming* in the optimization community (Best, 1982).

In a class of parametric quadratic programs (QPs), the solution path is represented as a piecewise-linear function of the problem parameters. If we regard the regularization parameter of the Support Vector Machine (SVM) as problem parameter, the optimization problem for the SVM is categorized in this class. Therefore, the SVM solutions are represented as piecewise-linear functions of the regularization parameter.

The solutions of these parametric QPs are characterized by active constraint set in the current solution. The linearity of the path comes from the fact that the Karush-Khun-Tucker (KKT) optimality conditions of these problems are represented as a linear system defined by the current active set, while the "piecewiseness" is the consequence of the changes in the active set. The piecewise-linear solution path algorithm repeatedly updates the linear system and active set. The point of active set change is called *breakpoint* in the literature. The path of solutions generated by this algorithm is very accurate and they satisfy the optimality conditions more strictly than other algorithms.

Many machine learning problems, however, do not require strict optimality of the solution. In fact, one of the popular SVM optimization algorithm, called sequential minimal optimization (SMO) (Platt, 1999), is known to produce suboptimal (approximated) solution, where the tolerance to the optimality (degree of approximated) can be specified by users. In many experimental studies, it has been demonstrated that the generalization performances of these suboptimal solutions are not significantly different from those of strictly optimal ones.

Therefore, the strict optimality of the solution path algorithm is often unnecessary. Furthermore, it adversely affects the computational efficiency of the algorithm. In fact, the solution path algorithm can be very slow when it encounters a large number of (seemingly redundant) breakpoints. Although some empirical studies suggest that the number of breakpoints grows linearly in the input size, in the worst case, it can grow exponentially (Gärtner et al., 2009). Another difficulty is in starting the solution path algo-

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

rithm from an approximated solution, for example obtained by SMO, because it does not satisfy the strict optimality requirement.

In order to address these issues in the current solution path algorithm, we introduce a *suboptimal solution path algorithm*. Our algorithm also generates piecewise-linear solution path, but the optimality tolerance (approximation level) can be arbitrary controlled by users. It allows to control the trade-off between the accuracy of the solution and the computational cost.

The presented suboptimal solution path algorithm has the following properties.

- First, the algorithm can reduce the number of breakpoints (which is the main computational bottleneck in solution path algorithm) by allowing multiple active set changes at one breakpoint. Although this modification causes what is called *degeneracy* problem, we provide an efficient and accurate way to solve this issue. We empirically show that reducing the number of breakpoints can work effectively to the computational efficiency.
- Second, the suboptimal solutions obtained by the algorithm can be interpreted as the solution of a *perturbed optimization problem* from the original one. This novel interpretation provides several insights into the properties of our suboptimal solutions. We present some theoretical analyses of our suboptimal solutions using this interpretation.

We also empirically investigate several practical properties of our approach. Although, our algorithm updates multiple active constraints at one breakpoint, we observe that the entire changing patterns of the active sets are very similar to those of the exact path. Moreover, despite its computational efficiency, the generalization performance of our suboptimal path is comparable to conventional one.

To the best of our knowledge, there are no previous works for suboptimal solution path algorithm with controllable optimality tolerance that can be applicable to standard SVM formulation [1]. Although many authors mimic the solution path by just repeating the warm-start on finely grid points (e.g., Friedman et al., 2007), this approach does not provide any guarantee about the intermediate solutions between grid points. In this paper we focus our attention to the solution path algorithm for standard SVM, but the presented approach can be applied to other problems in the aforementioned QP class.

[1] Giesen et al. (2010) proposed approximated path algorithm with some optimality guarantee that can be applicable to L2-SVM without bias term.

2. Solution Path for Support Vector Machine

In this section, we describe the solution path algorithm for regularization parameters of Support Vector Machine (SVM).

2.1. Support Vector Machine

Suppose we have a set of training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^p$ is the input and $y_i \in \{-1, +1\}$ is the output class label. SVM learns a linear discriminant function $f(\boldsymbol{x}) = \boldsymbol{w}^\top \Phi(\boldsymbol{x}) + \alpha_0$ in a feature space $\mathcal{F}$, where $\Phi : \mathcal{X} \to \mathcal{F}$ is a map from the input space $\mathcal{X}$ to the feature space $\mathcal{F}$, $\boldsymbol{w} \in \mathcal{F}$ is a coefficient vector and $\alpha_0 \in \mathbb{R}$ is a bias term.

In this paper, we consider the optimization problem of the following form:

$$\min_{\boldsymbol{w}, \alpha_0, \{\xi_i\}_{i=1}^n} \quad \tfrac{1}{2}\|\boldsymbol{w}\|_2^2 + \textstyle\sum_{i=1}^n C_i \xi_i, \qquad (1)$$
$$\text{s.t.} \;\; y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i, \; \xi_i \geq 0, \; i = 1, \ldots, n,$$

where $\{C_i\}_{i=1}^n$ denotes regularization parameters. This formulation reduces to the standard formulation of the SVM when all C_i's are the same. Our discussion in this paper holds for arbitrary choice of C_i's.

We formulate the dual problem of (1) as:

$$\max_{\boldsymbol{\alpha}} \; -\tfrac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q} \boldsymbol{\alpha} + \boldsymbol{1}^\top \boldsymbol{\alpha} \qquad (2)$$
$$\text{s.t.} \; \boldsymbol{y}^\top \boldsymbol{\alpha} = 0, \; \boldsymbol{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{c},$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top$, $\boldsymbol{c} = [C_1, \ldots, C_n]^\top$ and (i, j) element of $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ is $Q_{ij} = y_i y_j \Phi(\boldsymbol{x}_i)^\top \Phi(\boldsymbol{x}_j)$. Note that, we use inequalities between vectors as the element-wise inequality (i.e., $\boldsymbol{\alpha} \leq \boldsymbol{c} \Leftrightarrow \alpha_i \leq C_i$ for $i = 1, \ldots, n$). Using kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^\top \Phi(\boldsymbol{x}_j)$, discriminant function f is represented as: $f(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + \alpha_0$.

In what follows, the subscript by an index set such as $\boldsymbol{v}_{\mathcal{I}}$ for a vector $\boldsymbol{v} = [v_1, \cdots, v_n]^\top$ indicates a sub-vector of $\boldsymbol{v}$ whose elements are indexed by $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$. Similarly, the subscript by two index sets such as $\boldsymbol{M}_{\mathcal{I}_1, \mathcal{I}_2}$ for a matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ denotes a sub-matrix whose rows and columns are indexed by $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. The principal sub-matrix such as $\boldsymbol{M}_{\mathcal{I},\mathcal{I}}$ is abbreviated as $\boldsymbol{M}_{\mathcal{I}}$.

2.2. Solution Path Algorithm for SVM

In this paper, we consider the solution path with respect to the regularization parameter vector $\boldsymbol{c}$. To follow the path, we parametrized $\boldsymbol{c}$ in the following form:

$$\boldsymbol{c}^{(\theta)} = \boldsymbol{c}^{(0)} + \theta \boldsymbol{d},$$

where $\boldsymbol{c}^{(0)} = [C_1^{(0)}, \dots, C_n^{(0)}]^\top$ is some initial parameter, $\boldsymbol{d} = [d_1, \dots, d_n]^\top$ is a direction of the path and $\theta \geq 0$. We trace the change of the optimal solution of the SVM when θ increases from 0.

Let $\{\alpha_i^{(\theta)}\}_{i=0}^n$ be the optimal parameters and $\{f_i^{(\theta)}\}_{i=1}^n$ be the outputs $f(\boldsymbol{x}_i)$ at θ. The KKT optimality conditions are summarized as:

$$y_i f_i^{(\theta)} \geq 1, \quad \text{if} \quad \alpha_i^{(\theta)} = 0, \tag{3a}$$
$$y_i f_i^{(\theta)} = 1, \quad \text{if} \quad 0 < \alpha_i^{(\theta)} < C_i^{(\theta)}, \tag{3b}$$
$$y_i f_i^{(\theta)} \leq 1, \quad \text{if} \quad \alpha_i^{(\theta)} = C_i^{(\theta)}, \tag{3c}$$
$$\boldsymbol{y}^\top \boldsymbol{\alpha} = 0. \tag{3d}$$

We separate data points into three index sets $\mathcal{M}, \mathcal{O}, \mathcal{I} \subseteq \{1, \dots, n\}$ in such a way that these sets satisfy

$$i \in \mathcal{O} \quad \Rightarrow \quad y_i f_i^{(\theta)} \geq 1, \alpha_i^{(\theta)} = 0, \tag{4a}$$
$$i \in \mathcal{M} \quad \Rightarrow \quad y_i f_i^{(\theta)} = 1, \alpha_i^{(\theta)} \in [0, C_i], \tag{4b}$$
$$i \in \mathcal{I} \quad \Rightarrow \quad y_i f_i^{(\theta)} \leq 1, \alpha_i^{(\theta)} = C_i, \tag{4c}$$

and we denote these partitions altogether as $\pi := (\mathcal{O}, \mathcal{M}, \mathcal{I})$. If every data point belongs to one of the three index sets and equality (3d) holds, the KKT conditions (3) are satisfied. As long as these index sets are unchanged, we have analytical expression of the optimal solution in the form of $\alpha_i^{(\theta+\Delta\theta)} = \alpha_i^{(\theta)} + \Delta\theta\beta_i$, $i = 0, \dots, n$, where $\Delta\theta$ is the change of θ and $\{\beta_i\}_{i=0}^n$ are constants derived from sensitivity analysis theory:

Theorem 1. *Let $\pi = (\mathcal{O}, \mathcal{M}, \mathcal{I})$ be the partition at the optimal solution at θ and assume that $\boldsymbol{M} = \begin{bmatrix} 0 & \boldsymbol{y}_\mathcal{M}^\top \\ \boldsymbol{y}_\mathcal{M} & \boldsymbol{Q}_\mathcal{M} \end{bmatrix}$ is non-singular[2]. Then, as long as π is unchanged, $\{\beta_i\}_{i=0}^n$ is given by*

$$\begin{bmatrix} \beta_0 \\ \boldsymbol{\beta}_\mathcal{M} \end{bmatrix} = -\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{y}_\mathcal{I}^\top \\ \boldsymbol{Q}_{\mathcal{M},\mathcal{I}} \end{bmatrix} \boldsymbol{d}_\mathcal{I}, \ \boldsymbol{\beta}_\mathcal{O} = \boldsymbol{0}, \ \boldsymbol{\beta}_\mathcal{I} = \boldsymbol{d}_\mathcal{I}. \tag{5}$$

The proof is in the longer version of this paper (Karasuyama & Takeuchi, 2011). This theorem can be viewed as one of the specific forms of the sensitivity theorem (Fiacco, 1976). It can be derived from the KKT conditions (3) and the similar properties are repeatedly used in various solution path algorithms in machine learning (Hastie et al., 2004).

Using the above theorem, we can update the solution by $\alpha_i^{(\theta+\Delta\theta)} = \alpha_i^{(\theta)} + \Delta\theta\beta_i$ as long as π is unchanged. However, if we changes θ, the optimal partition π could also changes. Those change points are called *breakpoints*. In the solution path algorithm, the optimality conditions are always kept satisfied by precisely detecting the breakpoints and updating π properly.

[2]The invertibility of the matrix $\boldsymbol{M}$ is assured if and only if the submatrix $\boldsymbol{Q}_\mathcal{M}$ is positive definite in subspace $\{\boldsymbol{z} \in \mathbb{R}^{|\mathcal{M}|} \mid \boldsymbol{y}_\mathcal{M}^\top \boldsymbol{z} = 0\}$.

3. Suboptimal Solution Path

In this section, we develop a suboptimal solution path algorithm for the SVM, where the tolerance to the optimality conditions can be arbitrary controlled by users. The basic idea is to relax the KKT optimality conditions and allow multiple data points to move among the partition π at the same time. Note that it reduces the number of breakpoints and leads to the improvement in its computational efficiency: allowing us to control the balance between the accuracy of the solution and the computational cost.

3.1. Approximate Optimality Conditions

First, we relax the conditions (4) as

$$i \in \mathcal{O} \Rightarrow y_i f_i^{(\theta)} \geq 1 - \varepsilon_1, \alpha_i^{(\theta)} \in [-\varepsilon_2, 0], \tag{6a}$$
$$i \in \mathcal{M} \Rightarrow y_i f_i^{(\theta)} \in [1 - \varepsilon_1, 1 + \varepsilon_1], \alpha_i^{(\theta)} \in [-\varepsilon_2, C_i^{(\theta)} + \varepsilon_2], \tag{6b}$$
$$i \in \mathcal{I} \Rightarrow y_i f_i^{(\theta)} \leq 1 + \varepsilon_1, \alpha_i^{(\theta)} \in [C_i^{(\theta)}, C_i^{(\theta)} + \varepsilon_2], \tag{6c}$$

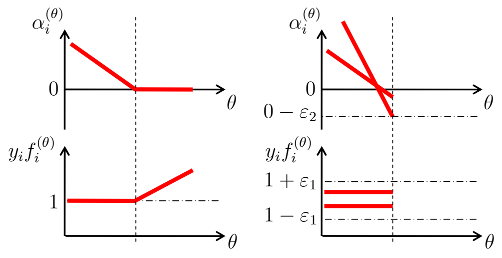
where $\varepsilon_1 \geq 0$ and $\varepsilon_2 \geq 0$ specify the degree of approximation. If we set $\varepsilon_1 = \varepsilon_2 = 0$, these conditions reduce to (4).

Our algorithm changes θ while keeping the above conditions (6) satisfied. Let $\theta_0 = 0$ be the initial value of θ and the non-decreasing sequence $\theta_0 \leq \theta_1 \leq \theta_2 \leq \dots$, be the breakpoints. Suppose we are currently at θ_k, the next breakpoint θ_{k+1} is characterized as the point that we can not increase θ without violating the conditions (6) or changing index sets π.

If we set $\{\beta_i\}_{i=0}^n$ by (5), then $y_i f_i^{(\theta)}$, $i \in \mathcal{M}$, and $\alpha_i^{(\theta)}$, $i \in \mathcal{O} \cup \mathcal{I}$, are constants. To increase θ from θ_k, we only need to check the following inequalities:

$$\begin{array}{rcll} y_i f_i^{(\theta_k)} + \Delta\theta g_i & \geq & 1 - \varepsilon_1, & i \in \mathcal{O}, \\ \alpha_i^{(\theta_k)} + \Delta\theta\beta_i & \in & [-\varepsilon_2, C_i^{(\theta_k)} + \varepsilon_2], & i \in \mathcal{M}, \\ y_i f_i^{(\theta_k)} + \Delta\theta g_i & \leq & 1 - \varepsilon_1, & i \in \mathcal{I}, \end{array}$$

where g_i is the change of output $y_i f_i$ which is defined by $\boldsymbol{g} = \boldsymbol{Q}\boldsymbol{\beta} + \boldsymbol{y}\beta_0$. We want to know the maximum $\Delta\theta$ which satisfies all of the above inequalities. We can easily calculate the maximum $\Delta\theta$ for each inequality



(a) Exact path (b) Suboptimal path

Figure 1. An illustrative example of the breakpoint. The points of the vertical dashed lines are breakpoints. (a) At the breakpoint in the upper plot, $\alpha_i, i \in \mathcal{M}$, reaches 0. Since the index i is transferred from $\mathcal{M}$ to $\mathcal{O}$, $\alpha_i = 0$ on the right side of the vertical line. In the lower plot, $y_i f_i^{(\theta)} = 1$ on the left side of the vertical line and $y_i f_i^{(\theta)} \geq 1$ on the right side of the vertical line. At the breakpoint, the data point i satisfies the both of the optimality conditions (4b) and (4a) for $\mathcal{M}$ and $\mathcal{O}$, respectively. (b) At the breakpoint in the upper plot, one of $\alpha_i, i \in \mathcal{M}$, reaches $-\varepsilon_2$. In the lower plot, both of the two lines are in $[1-\varepsilon_1, 1+\varepsilon_1]$. In this case, these two points satisfy the both of the optimality conditions (6b) and (6a) for $\mathcal{M}$ and $\mathcal{O}$, respectively. It does not necessarily mean that these two data points should move to $\mathcal{O}$: either of them have a possibility to stay in $\mathcal{M}$ even after the breakpoint. This situation is called degeneracy in parametric programming literature.

as follows:

$$\begin{aligned}
\Theta_{\mathcal{O}} &= \left\{(1-\varepsilon_1 - y_i f_i^{(\theta_k)})/g_i \middle| i \in \mathcal{O}, g_i < 0\right\},\\
\Theta_{\mathcal{M}_\ell} &= \left\{-(\alpha_i^{(\theta_k)} + \varepsilon_2)/\beta_i \middle| i \in \mathcal{M}, \beta_i < 0\right\},\\
\Theta_{\mathcal{M}_u} &= \Big\{(C_i^{(\theta_k)} + \varepsilon_2 - \alpha_i^{(\theta_k)})/(\beta_i - d_i)\\
&\qquad\qquad \Big| i \in \mathcal{M}, \beta_i > d_i\Big\},\\
\Theta_{\mathcal{I}} &= \left\{(1+\varepsilon_1 - y_i f_i^{(\theta_k)})/g_i \middle| i \in \mathcal{I}, g_i > 0\right\},
\end{aligned}$$

Since we have to keep all of the inequalities satisfied, we take the minimum of these values: $\Delta\theta = \min \Theta$, where $\Theta = \{\Theta_{\mathcal{O}}, \Theta_{\mathcal{M}_\ell}, \Theta_{\mathcal{M}_u}, \Theta_{\mathcal{I}}\}$. Then we can find $\theta_{k+1} = \theta_k + \Delta\theta$.

Although we detect θ_{k+1}, it is necessary to update π to go beyond the breakpoint. Conventional solution path algorithms allow only one data point to move between the partition π at each breakpoint. For example, α_i, $i \in \mathcal{M}$, reaches 0, the algorithm transfers the index i from $\mathcal{M}$ to $\mathcal{O}$ (Figure 1(a)). In our algorithm, multiple data points are allowed to move between the partitions π at the same time in order to reduce the number of breakpoints.

3.2. Update Index Sets

At a breakpoint, our algorithm handles all the data points that violate the strict inequality conditions (4) rather than the relaxed ones (6) (Figure 1(b)). This situation can be interpreted as what is called *degeneracy* in the parametric programming (Ritter, 1984). Here, degeneracy means that multiple constraints hit their boundaries of inequalities simultaneously. Although degenerate situation rarely happens in conventional solution path algorithms, it is not the case in ours. The simultaneous change of multiple data points inevitably brings about "highly" degenerate situations involved with many constraints. In degenerate case, we have a problem called the *cycling*. For example, if we move two indices i and j from $\mathcal{M}$ to $\mathcal{O}$ at the breakpoint, then both or either of them may immediately return to $\mathcal{M}$. To avoid the cycling, we need to design an update strategy for π that can circumvent cycling.

The degeneracy can be handled by several approaches which are known in the parametric programming literature. Ritter (1984) showed that the cycling can be dealt with through the well-known Bland's minimum index rule in the linear programming (Bland, 1977). However, in the worst case, this approach must go through all the possible patterns of next π. Since we need to evaluate $\{\beta_i\}_{i=0}^n$ in each iteration, a large number of iterations may cause additional computational cost. In this paper, we provide more essential solution to this problem based on (Berkelaar et al., 1997).

Suppose we are currently on the breakpoint θ_k. Let

$$\begin{aligned}
\mathcal{B}_{\mathcal{O}} &= \{i \mid \alpha_i^{(\theta_k)} \leq 0, \beta_i < 0, i \in \mathcal{M}\} \cup\\
&\quad\ \{i \mid y_i f_i^{(\theta_k)} \leq 1, g_i < 0, i \in \mathcal{O}\},\\
\mathcal{B}_{\mathcal{I}} &= \{i \mid \alpha_i^{(\theta_k)} \geq C_i^{(\theta_k)}, \beta_i > d_i, i \in \mathcal{M}\} \cup\\
&\quad\ \{i \mid y_i f_i^{(\theta_k)} \geq 1, g_i > 0, i \in \mathcal{I}\}.
\end{aligned}$$

$\mathcal{B}_{\mathcal{O}}$ is the set of indices which satisfy the conditions (6a) and (6b) for being the member of $\mathcal{M}$ and $\mathcal{O}$ simultaneously at θ_k. Similarly, indices in $\mathcal{B}_{\mathcal{I}}$ satisfy the conditions (6b) and (6c) for being the member of $\mathcal{M}$ and $\mathcal{I}$ at θ_k. Moreover, let us define sum of these two sets as $\mathcal{B} = \mathcal{B}_{\mathcal{O}} \cup \mathcal{B}_{\mathcal{I}}$. Our task is to partition these indices to $\mathcal{O}$, $\mathcal{M}$ and $\mathcal{I}$ correctly so that it does not cause the cycling.

In our formulation, due to the approximation by ε_1 and ε_2, the cycling may not occur at $\Delta\theta = 0$ immediately. For example, suppose that i move to $\mathcal{M}$ from $\mathcal{O}$ and its parameter is $\alpha_i = 0$. In the next iteration, we need to check $\alpha_i + \Delta\theta\beta_i \geq -\varepsilon_2$. If $\beta_i < 0$, then we obtain $\Delta\theta \leq -\varepsilon_2/\beta_i > 0$. Although it allows $\Delta\theta > 0$, the index i may return back to $\mathcal{O}$. This situation can also be considered as cycling.

Let $\pi_k = (\mathcal{O}_k, \mathcal{M}_k, \mathcal{I}_k)$ be π in $[\theta_k, \theta_{k+1}]$. At θ_{k+1}, if and only if the cycling does not occur, it can be shown that the following conditions hold:

$$\beta_i \geq 0, g_i = 0, \quad \text{for } i \in \mathcal{M}_{k+1} \cap \mathcal{B}_{\mathcal{O}}, \tag{7a}$$
$$\beta_i = 0, g_i \geq 0, \quad \text{for } i \in \mathcal{O}_{k+1} \cap \mathcal{B}_{\mathcal{O}}, \tag{7b}$$
$$\beta_i \leq d_i, g_i = 0, \quad \text{for } i \in \mathcal{M}_{k+1} \cap \mathcal{B}_{\mathcal{I}}, \tag{7c}$$
$$\beta_i = d_i, g_i \leq 0, \quad \text{for } i \in \mathcal{I}_{k+1} \cap \mathcal{B}_{\mathcal{I}}. \tag{7d}$$

Although β_i and g_i are usually calculated using π, our approach allows us to calculate β_i and g_i without knowing π so that they can satisfy the above conditions. If the gradient $\boldsymbol{\beta}$, which is defined in (5), satisfies the following conditions, we can find the next partition π_{k+1} to satisfy (7). The conditions are:

$$\begin{aligned} g_i\beta_i = 0,\ g_i \geq 0,\ \beta_i \geq 0,\ i \in \mathcal{B}_{\mathcal{O}}, \\ g_i(d_i - \beta_i) = 0,\ g_i \leq 0,\ \beta_i \leq d_i,\ i \in \mathcal{B}_{\mathcal{I}}, \end{aligned} \tag{8}$$

If we know such $\boldsymbol{\beta}$ and $\boldsymbol{g}$, using the following update rule, we can determine π_{k+1} as:

$$\begin{aligned} \mathcal{M}_k = \ & \mathcal{M}_{k+\frac{1}{2}} \cup \{i \mid \beta_i > 0, g_i = 0, i \in \mathcal{B}_{\mathcal{O}}\} \\ & \cup \{i \mid \beta_i < d_i, g_i = 0, i \in \mathcal{B}_{\mathcal{I}}\}, \\ \mathcal{O}_k = \ & \mathcal{O}_{k+\frac{1}{2}} \cup \{i \mid \beta_i = 0, g_i \geq 0, i \in \mathcal{B}_{\mathcal{O}}\}, \\ \mathcal{I}_k = \ & \mathcal{I}_{k+\frac{1}{2}} \cup \{i \mid \beta_i = d_i, g_i \leq 0, i \in \mathcal{B}_{\mathcal{I}}\}, \end{aligned} \tag{9}$$

where $\mathcal{O}_{k+\frac{1}{2}} = \mathcal{O}_k \setminus \mathcal{B}$, $\mathcal{M}_{k+\frac{1}{2}} = \mathcal{M}_k \setminus \mathcal{B}$ and $\mathcal{I}_{k+\frac{1}{2}} = \mathcal{I}_k \setminus \mathcal{B}$.

Remark 1. *By definition, the update rule (9) guarantees that the non-cycling conditions (7) hold.*

To use (9), we need $\boldsymbol{\beta}$ (5) which satisfies (8). The following theorem shows that it can be obtained from a quadratic programming problem (QP):

Theorem 2. *Let $\widehat{\beta}_0$, $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{g}}$ be the optimal solutions of the following QP problem:*

$$\min_{\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{g}}} \sum_{i \in \mathcal{B}_{\mathcal{O}}} \widehat{g}_i \widehat{\beta}_i + \sum_{i \in \mathcal{B}_{\mathcal{I}}} \widehat{g}_i (\widehat{\beta}_i - d_i) \tag{10}$$
$$\text{s.t.} \begin{cases} \widehat{\boldsymbol{g}}_{\mathcal{B}_{\mathcal{O}}} \geq \boldsymbol{0},\ \widehat{\boldsymbol{\beta}}_{\mathcal{B}_{\mathcal{O}}} \geq \boldsymbol{0},\ \widehat{\boldsymbol{g}}_{\mathcal{B}_{\mathcal{I}}} \leq \boldsymbol{0},\ \widehat{\boldsymbol{\beta}}_{\mathcal{B}_{\mathcal{I}}} \leq \boldsymbol{d}_{\mathcal{I}}, \\ \widehat{\boldsymbol{g}}_{\mathcal{M}_{k+\frac{1}{2}}} = \boldsymbol{0},\ \widehat{\boldsymbol{\beta}}_{\mathcal{O}_{k+\frac{1}{2}}} = \boldsymbol{0},\ \widehat{\boldsymbol{\beta}}_{\mathcal{I}_{k+\frac{1}{2}}} = \boldsymbol{d}_{\mathcal{I}_{k+\frac{1}{2}}}, \\ \boldsymbol{y}^\top \widehat{\boldsymbol{\beta}} = 0,\ \widehat{\boldsymbol{g}} = \boldsymbol{Q}\widehat{\boldsymbol{\beta}} + \boldsymbol{y}\widehat{\beta}_0, \end{cases}$$

and π is determined by (9) using $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{g}}$. Then $\widehat{\beta}_0$, $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{g}}$ satisfy (8) and they are equal to the gradient β_0, $\boldsymbol{\beta}$ and $\boldsymbol{g}$, respectively.

Although the detailed proof is in (Karasuyama & Takeuchi, 2011), we can provide clear interpretation of this optimization problem. The objective function and inequality constraints corresponds to (8) and the other constraints correspond to the linear system (5). It can be shown that the optimal value of the objective function is 0. Given the non-negativity of each term in the objective, we see that (8) holds (see Karasuyama & Takeuchi, 2011, for detail).

The optimization problem (10) has $2n+1$ variables and $2|\mathcal{B}|+2n+1$ constraints. However, we can reduce these sizes to $|\mathcal{B}|$ variables and $2|\mathcal{B}|$ constraints by arranging the equality constraints[3]. The detailed formulation of the reduced problem is in (Karasuyama & Takeuchi, 2011). If the size of $|\mathcal{B}|$ is large, it may take large computational cost to solve (10). To avoid this, we set the upper bound B for the number of elements of $\mathcal{B}$. In the case of $|\mathcal{B}| > B$, we choose top B elements from the original $\mathcal{B}$ by increasing order of Θ as the elements of $\mathcal{B}$.

3.3. Algorithm and Computational Complexity

Here, we summarize our algorithm and analyze its computational complexity. At the k-th breakpoint, our algorithm performs the following procedure:

step1 Using π_k, calculate $\beta_0, \boldsymbol{\beta}$ and $\boldsymbol{g}$ by (5)

step2 Calculate the next breakpoint θ_{k+1} and update $\alpha_0^{(\theta)}, \boldsymbol{\alpha}^{(\theta)}, \boldsymbol{c}^{(\theta)}$;

step3 Solve (10) and calculate π_{k+1} by (9)

In step1, we need to solve the linear system (5). In conventional solution path algorithms, we can update it using rank-one-update of an inverse matrix or a Cholesky factor from previous iteration by $O(|\mathcal{M}|^2)$ computations. In our case, we need rank-m-update at each breakpoint, where $1 \leq m \leq B$. When we set B as some small constant, the computational cost still remains $O(|\mathcal{M}|^2)$. Including the other processes in this step, the computational cost becomes $O(n|\mathcal{M}|)$. In step2, given $\boldsymbol{\beta}$ and $\boldsymbol{g}$, we can calculate all the possible step length Θ by $O(n)$. In step3, since the optimization problem (10) becomes convex QP problem with $|\mathcal{B}|$ variables, it can be solved efficiently by some standard QP solvers in the situation $|\mathcal{B}|$ is relatively small compared to n. When we set B as some constant, the time for solving this optimization problem is then independent of n.

Put it all together, in the case of constant B, the computational cost of each breakpoint is $O(n|\mathcal{M}|)$. This is the same as the conventional solution path algorithm. However, as we will see later in experiments, our al-

[3]In the case of $|\mathcal{M}_{k+\frac{1}{2}}| = 0$, the reduced problem has $|\mathcal{B}| + 1$ variables $2|\mathcal{B}| + 1$ constraints.

gorithm drastically reduces the number of breakpoints especially when we use large ε_1 and ε_2.

4. Analysis

In this section, we provide some theoretical analyses of our suboptimal solution path.

4.1. Interpretation as Perturbed Problem

An interesting property of our approach is that the solutions always keep the optimality of an optimization problem which is slightly perturbed from the original one. The following theorem gives the formulation of the perturbed problem:

Theorem 3. *Every solution $\boldsymbol{\alpha}^{(\theta)}$ in the suboptimal solution path is the optimal solution of the following optimization problem:*

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\tfrac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} + (\mathbf{1}+\boldsymbol{p})^\top \boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{y}^\top\boldsymbol{\alpha} = 0,\ -\boldsymbol{q} \le \boldsymbol{\alpha} \le \boldsymbol{c}^{(\theta)} + \boldsymbol{q}. \end{aligned} \tag{11}$$

where perturbation parameters $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n$ are in $-\varepsilon_1\mathbf{1} \le \boldsymbol{p} \le \varepsilon_1\mathbf{1}$ and $\mathbf{0} \le \boldsymbol{q} \le \varepsilon_2\mathbf{1}$, respectively.

Proof. Let $\boldsymbol{\xi}^+, \boldsymbol{\xi}^- \in \mathbb{R}^n_+$ and $\kappa \in \mathbb{R}$ be the Lagrange multipliers. The Lagrangian is

$$\begin{aligned} L \;=\; & -\tfrac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} + (\mathbf{1}+\boldsymbol{p})^\top \boldsymbol{\alpha} \\ & +(\boldsymbol{\alpha}+\boldsymbol{q})^\top\boldsymbol{\xi}^- + (\boldsymbol{c}^{(\theta)} + \boldsymbol{q} - \boldsymbol{\alpha})^\top \boldsymbol{\xi}^+ + \kappa \boldsymbol{y}^\top\boldsymbol{\alpha}, \end{aligned}$$

and the KKT conditions are

$$\frac{\partial L}{\partial \boldsymbol{\alpha}} = -\boldsymbol{Q}\boldsymbol{\alpha} + \mathbf{1} + \boldsymbol{p} + \boldsymbol{\xi}^- - \boldsymbol{\xi}^+ + \kappa\boldsymbol{y} = \mathbf{0}, \tag{12a}$$

$$\boldsymbol{\xi}^+, \boldsymbol{\xi}^- \ge 0, \tag{12b}$$

$$\xi_i^-(\alpha_i + q_i) = 0,\ i = 1,\ldots,n, \tag{12c}$$

$$\xi_i^+(C_i^{(\theta)} + q_i - \alpha_i) = 0,\ i = 1,\ldots,n, \tag{12d}$$

$$-\boldsymbol{q} \le \boldsymbol{\alpha} \le \boldsymbol{c}^{(\theta)} + \boldsymbol{q}. \tag{12e}$$

$$\boldsymbol{y}^\top\boldsymbol{\alpha} = 0, \tag{12f}$$

Substituting $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{(\theta)}$ and $\kappa = -\alpha_0^{(\theta)}$, i-th element of (12a) can be written as $y_i f_i^{(\theta)} = 1 + p_i + \xi_i^- - \xi_i^+$. Considering this and the conditions of suboptimal solution $\boldsymbol{\alpha}^{(\theta)}$ (6), there exist $p_i \in [-\varepsilon_1, \varepsilon_1]$ and $\xi_i^{\pm}$ which satisfy $\xi_i^+ = \xi_i^- = 0$ for $i \in \mathcal{M}$, $\xi_i^+ = 0,\ \xi_i^- \ge 0$, for $i \in \mathcal{O}$ and $\xi_i^+ \ge 0,\ \xi_i^- = 0$, for $i \in \mathcal{I}$. These $\xi_i^{\pm}$'s satisfy the non-negativity constraint (12b).

The complementary conditions (12c) and (12d) for $i \in \mathcal{M}$ hold from $\xi_i^+ = \xi_i^- = 0$. For $i \in \mathcal{O}$, since $\xi_i^+ = 0$, we don't have to check (12d). In this case, if we set $q_i = -\alpha_i^{(\theta)} \in [0, \varepsilon_2]$, then (12c) holds. It can be shown in a similar way that (12c) and (12d) hold for $i \in \mathcal{I}$.

Our suboptimal solution path algorithm always satisfies the equality constraint of the dual (2) and the box constraint (12e) satisfied. Therefore, we see (12) holds. $\square$

The problem (11) can be interpreted as the dual problem of the following form of the SVM:

$$\min_{\boldsymbol{w}, \alpha_0}\ \tfrac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + \textstyle\sum_{i=1}^n \ell(1 + p_i - y_i f_i), \tag{13}$$

where

$$\ell(\xi_i) = \begin{cases} (C_i^{(\theta)} + q_i)\xi_i, & \text{for } \xi_i \ge 0, \\ -q_i\xi_i, & \text{for } \xi_i < 0, \end{cases}$$

is a loss function. We see that the perturbations present in the loss term.

4.2. Error Analysis

We have shown that the solution of the suboptimal solution path can be interpreted as the optimal solution of the perturbed problem (13). Here, we consider how close the optimal solution of the perturbed problem to the solution of the original problem in terms of the optimal objective value.

Let $D(\boldsymbol{\alpha})$ and $\widetilde{D}(\boldsymbol{\alpha})$ be the dual objective functions of the original optimization problem (2) and the perturbed problem (11), respectively. From the affine lower bound of $\widetilde{D}(\boldsymbol{\alpha})$, we obtain

$$\widetilde{D}(\boldsymbol{\alpha}) \le D(\boldsymbol{\alpha}^*) + \boldsymbol{p}^\top\boldsymbol{\alpha}^* + (-\boldsymbol{Q}\boldsymbol{\alpha}^* + \mathbf{1} + \boldsymbol{p})^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*),$$

where $\boldsymbol{\alpha}^*$ is the optimal solution of the original problem. Let $\widetilde{\boldsymbol{\alpha}}$ be the optimal solution of the perturbed problem. Substituting $\boldsymbol{\alpha} = \widetilde{\boldsymbol{\alpha}}$ and adding $\alpha_0^*\boldsymbol{y}^\top(\widetilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*) = 0$ to the right hand side, we obtain

$$\widetilde{D}(\widetilde{\boldsymbol{\alpha}}) - D(\boldsymbol{\alpha}^*) \le \boldsymbol{p}^\top\boldsymbol{\alpha}^* + (\boldsymbol{\xi}^* + \boldsymbol{p})^\top(\widetilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*), \tag{14}$$

where $\boldsymbol{\xi}^* = -\boldsymbol{Q}\boldsymbol{\alpha}^* - \boldsymbol{y}\alpha_0^* + \mathbf{1}$. Note that $\boldsymbol{\xi}^*_{\mathcal{I}} \ge \mathbf{0}$, $\boldsymbol{\xi}^*_{\mathcal{M}} = \mathbf{0}$ and $\boldsymbol{\xi}^*_{\mathcal{O}} \le \mathbf{0}$, where $\mathcal{I}$, $\mathcal{M}$ and $\mathcal{O}$ represent the optimal partition of the original problem (2). Here, we define $\widetilde{\mathcal{I}} = \{i \mid \xi_i^* + p_i \ge 0,\ i \in \mathcal{I}\}$, $\widetilde{\mathcal{O}} = \{i \mid \xi_i^* + p_i \le 0,\ i \in \mathcal{O}\}$ and $\widetilde{\mathcal{M}} = \{1,\ldots,n\} \setminus (\widetilde{\mathcal{O}} \cup \widetilde{\mathcal{I}})$. From the right hand side of (14), we obtain

$$\begin{aligned} \widetilde{D}(\widetilde{\boldsymbol{\alpha}}) - D(\boldsymbol{\alpha}^*) \le & \textstyle\sum_{i \in \mathcal{M}\cup\mathcal{I}} |p_i|\ C_i^{(\theta)} + \\ & \textstyle\sum_{i \in \widetilde{\mathcal{I}}\cup\widetilde{\mathcal{O}}} |\xi_i^* + p_i|\ q_i + \sum_{i \in \widetilde{\mathcal{M}}} |p_i|\ (C_i^{(\theta)} + q_i) \end{aligned}$$

From the duality theorem, this also bounds the difference of the primal objective value. Comparing the original objective function (1), this bound can be considered small when p_i and q_i is enough small compared to ξ_i^* and C_i. In this view point, this bound gives theoretical justification for our intuitive interpretation. The bound for $D(\boldsymbol{\alpha}^*) - \widetilde{D}(\widetilde{\boldsymbol{\alpha}})$ can be also derived in the same manner.

5. Experiments

In this section, we illustrate the empirical performance of the proposed approach compared to the conventional exact solution path algorithm. Our task is to trace the solution path from $\boldsymbol{c}^{(0)} = 10^{-1}/n \times \mathbf{1}$ to $\boldsymbol{c}^{(1)} = 10^{6}/n \times \mathbf{1}$. Since all the elements of $\boldsymbol{c}^{(\theta)}$ takes the same value in this case, we sometimes refer to this common value as $C^{(\theta)}$ (i.e., $\boldsymbol{c}^{(\theta)} = C^{(\theta)} \times \mathbf{1}$). The RBF kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2)$ is used with $\gamma = 1/p$ where p is the number of features. To circumvent possible numerical instability in the solution path, we add small positive constant 10^{-6} to the diagonals of the matrix $\boldsymbol{Q}$.

Let $e \geq 0$ be a parameter which controls the degree of approximations. In this paper, using e, we set ε_1 and ε_2 as $\varepsilon_1 = e$ and $\varepsilon_2 = e \times C^{(\theta_k)}$, respectively, where θ_k is the previous breakpoint. We set ε_2 using relative scale to $C^{(\theta_k)}$.

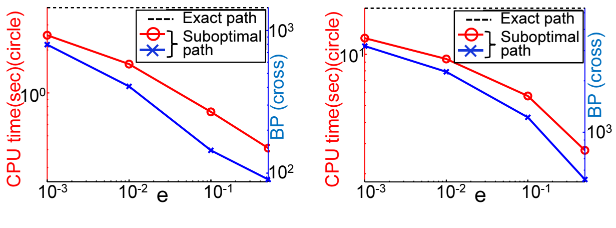
We used the following four data sets for comparison: 'internet ad', 'spam', 'a5a' and 'w5a'. The size of each data set is $(n, p) = (2539, 1558), (4601, 57), (6414, 123)$ and $(9888, 300)$, respectively. These data sets are available from LIBSVM site[4] and UCI data repository[5]. We randomly sampled approximately 80% data points from the original data set 10 times. The input $\boldsymbol{x}$ of each data set is linearly scaled to $[0, 1]^p$.

Figure 2 shows the comparison of the CPU time and the number of breakpoints. To make fair comparison, the initialization is not included in the CPU time. In these results, we set $B = 10$ and we investigated the relationship between the computational cost and the degree of approximation by examining several settings of $e \in \{0.001, 0.01, 0.1, 0.5\}$. The results indicate that our approach can reduce the CPU time especially when e is large. The number of breakpoints were also reduced, in the same way as the CPU time. In our approach, since we need rank-m-update of matrix in each breakpoint $(1 \leq m \leq B)$, an update in a breakpoint may take longer time than rank-one-update which is needed in the conventional solution path algorithm. We conjecture that this is why the decrease in the number of breakpoints was slightly faster than the CPU time. However, since the maximum value of $|\mathcal{B}|$ was set as $B = 10$ in this experiment, this additional cost was relatively small compared to the effect of the reduction of the number of breakpoints.

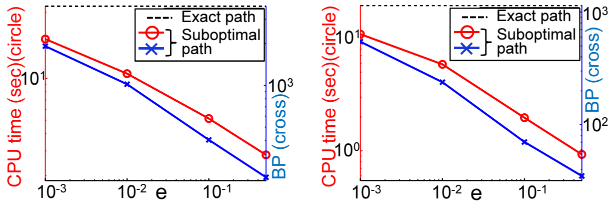
Next, we investigated the effect of B. Figure 3 shows the CPU time and the number of breakpoints for w1a data ($n = 2477$, $p = 300$) with $B = 10$ and $B = n$. When $B = n$, there are no upper bounds for $|\mathcal{B}|$. In the left plot, when $B = n$, we see that the CPU time is longer than the case of $B = 10$. In this data set, this difference of the CPU time mainly comes from the cost of the matrix update and QP (10) whose size is proportional to $|\mathcal{B}|$ (data not shown). On the other hand, in the left plot, the number of breakpoints is stable in the both case of $B = n$ and $B = 10$, and interestingly, the number itself is almost the same in these two settings. Our results suggest that too many $\mathcal{B}$ does not contribute to reduce the number of breakpoint. Although these unstable results in $B = n$ is not always happen, we observed that it is more stable to use $B = 10$ or $B = 100$ in several other data sets.

[4]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[5]http://archive.ics.uci.edu/ml/

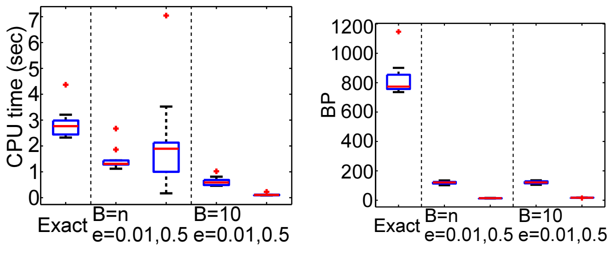


(a) ad (b) spam



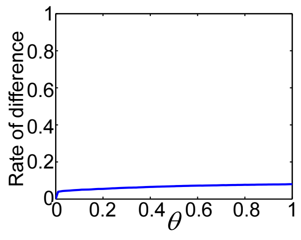
(c) a5a (d) w5a

Figure 2. Log plot of CPU time and the number of breakpoints (BP). The horizontal axis of each plot is the degree of the approximation. The circle denotes the CPU time (left axis) and the cross mark denotes the number of breakpoints (right axis) of the suboptimal path. The top dashed line of each plot means both of the CPU time and the number of breakpoints of the exact path. The relative scale of the left and right axes are the same.

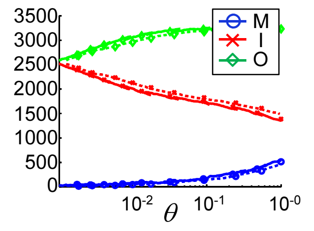


(a) CPU time (b) Breakpoint (BP)

Figure 3. The comparisons for different settings of B.



(a) The difference of π



(b) The size of each index set (solid: exact path, dashed: suboptimal path)

Figure 4. Comparisons of the behavior of π

Table 1. Test error rate and its standard error

data	exact path	$e = 0.5$
ad	0.0326 (0.0021)	0.0328 (0.0026)
spam	0.0770 (0.0036)	0.0812 (0.0037)
a5a	0.1587 (0.0025)	0.1597 (0.0031)
w5a	0.0171 (0.0012)	0.0176 (0.0010)

We also compared the difference of π between the exact solution path and the suboptimal path in order to see the degree of approximation in terms of the active set. Let $I_i \in \{0, 1\}$ be an indicator variable which has 1 when a data point i belongs to different set among $\mathcal{M}$, $\mathcal{O}$ and $\mathcal{I}$ between two solution paths. Figure 4(a) shows plots of 10 runs average of $\sum_{i=1}^{n} I_i/n$ for $e = 0.5$ in a5a data set. We see that the difference is at most about 10%. Figure 4(b) shows the size of each index set (this plot is one of 10 runs). Although the small differences exist, the changing patterns are similar each other.

Table 1 shows results of test error rate comparison for $e = 0.5$. We used 60% of the data for training, 20% for validation and 20% for testing. In each data set, we see that the performances of our suboptimal solutions are comparable to the exact solution path.

6. Conclusion

In this paper, we have developed a suboptimal solution path algorithm which traces the changes of solutions under the relaxed optimality conditions. Our algorithm can reduce the number of breakpoints by moving multiple indices in π at one breakpoint. Another interesting property of our approach is that the suboptimal solutions exactly correspond to the optimal solutions of the perturbed problems from the original SVM optimization problems. The experimental results demonstrate that our algorithm efficiently follows the path and it has similar patterns of active sets and classification performances compared to the exact path.

References

Berkelaar, A. B., Roos, K., and Terláky, T. The optimal set and optimal partition approach to linear and quadratic programming. In Greenberg, H. and Gal, T. (eds.), *Advances in Sensitivity Analysis and Parametric Programming*, chapter 6. Kluwer Academic Publishers, 1997.

Best, M. J. An algorithm for the solution of the parametric quadratic programming problem. Technical Report 82-24, Faculty of Mathematics, University of Waterloo, 1982.

Bland, R. G. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2:103–107, 1977.

Efron, B., Hastie, T., Johnstone, L., and Tibshirani, R. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

Fiacco, A. V. Sensitivity analysis for nonlinear programming using penalty methods. *Mathematical Programming*, 10(3):287–311, 1976.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

Gärtner, B., Giesen, J., and Jaggi, M. An exponential lower bound on the complexity of regularization paths. *arXiv:0903.4817v2 [cs.LG]*, 2009.

Giesen, J., Jaggi, M., and Laue, S. Approximating parameterized convex optimization problems. In de Berg, Mark and Meyer, Ulrich (eds.), *18th European Symposium on Algorithms*, volume 6346 of *Lecture Notes in Computer Science*, pp. 524–535. Springer Berlin / Heidelberg, 2010.

Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

Karasuyama, M. and Takeuchi, I. Suboptimal solution path algorithm for support vector machine. *arXiv:1105.0471 [cs.LG]*, 2011.

Platt, J. C. Fast training of support vector machines using sequential minimal optimization. In Schölkopf, Bernhard, Burges, Christopher J. C., and Smola, Alexander J. (eds.), *Advances in Kernel Methods — Support Vector Learning*, pp. 185–208, Cambridge, MA, 1999. MIT Press.

Ritter, K. On parametric linear and quadratic programming problems. In Cottle, R., Kelmanson, M. L., and Korte, B. (eds.), *Mathematical Programming: Proceedings of the International Congress on Mathematical Programming*, pp. 307–335. Elsevier Science Publisher B.V., 1984.