Pattern Recognition Letters 100 (2017) 51–58

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Effective subset approach for SVMpath singularities

Jisheng Dai[a], Weichao Xu[b], Zhongfu Ye[c], Chunqi Chang[d,*]

[a] Department of Electronic Engineering, Jiangsu University, 301 Xuefu Road, Zhenjiang 212013, China
[b] Department of Automatic Control, Guangdong University of Technology, 100 Huanxi Road, Guangzhou 510006, China
[c] Department of Electronic Engineering and Information Science, University of Science and Technology of China, 443 Huangshan Road, Hefei 230027, China
[d] School of Biomedical Engineering, Shenzhen University Health Science Center, Shenzhen 518060, China

A R T I C L E   I N F O

Article history:
Received 25 November 2015
Available online 5 September 2017

Keywords:
Support vector machine (SVM)
Solution path
Homotopy method
Singularity

A B S T R A C T

Singularities are frequently encountered in the powerful SVMpath algorithm. This paper proposes an effective subset approach for handling singularities, which is quite distinct from the previous ridge-adding method that can only obtain some approximate solutions. The main novelty of the new approach is to divide the active set into the effective and ineffective subsets. The effective subset plays an important role in guaranteeing the existence of the inverse matrix, so as to obtain a correct search direction even some singularities are encountered; while the ineffective subset is to temporarily absorb any indexes that might cause singularities, and properly release them into the effective subset when it is necessary. Experimental results are performed to verify our theoretical analysis and illustrate the ability of singularity avoidance.

1. Introduction

Support vector machine (SVM) has attracted tremendous interest as a popular tool for pattern classification, e.g., [1,2], due to its excellent generalization performance on a wide range of learning problems. Training a typical binary SVM requires solving a quadratic programming (QP) problem. In particular, given a set of training data $\{\mathbf{x}_i,\ y_i\}$, $i = 1, 2, \ldots, N$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the feature vector of the ith observation and $y_i \in \{-1, +1\}$ is its label, the standard formulation for a typical binary SVM classification problem is

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\varepsilon}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\varepsilon_i \\ &\text{s.t. } y_i(\mathbf{w}^T\phi(\mathbf{x}_i) - b) \geq 1 - \varepsilon_i, \quad \forall i \\ &\qquad \varepsilon_i \geq 0, \quad \forall i \end{aligned} \tag{1}$$

where $C > 0$ is a cost parameter that controls the trade-off between the regularization effect and the empirical risk minimization, $\phi : \mathbb{R}^n \mapsto \mathcal{H}$ is a function that maps the feature points $\mathbf{x}_i$ onto a high-dimensional Hilbert space $\mathcal{H}$. Within this paper, boldface upper-case letters denote matrices, boldface lower-case letters denote column vectors, and italics denote scalars. $(\cdot)^T$ denotes the transpose of a vector or a matrix, and a_i denotes the ith component of vector $\mathbf{a}$.

We can construct the Lagrangian dual form of (1) as:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{N}\alpha_i \\ &\text{s.t. } \sum_{i=1}^{N}\alpha_i y_i = 0 \\ &\qquad 0 \leq \alpha_i \leq C, \quad \forall i \end{aligned} \tag{2}$$

or, equivalently

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} -\frac{1}{2\lambda}\boldsymbol{\alpha}^T\mathbf{D}^T\mathbf{D}\boldsymbol{\alpha} + \mathbf{1}^T\boldsymbol{\alpha} \\ &\text{s.t. } \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ &\qquad 0 \leq \boldsymbol{\alpha} \leq \mathbf{1} \end{aligned} \tag{3}$$

where α_is are the Lagrange multipliers, $K(\mathbf{x}_i,\ \mathbf{x}_j) \triangleq <\phi(\mathbf{x}_i),\ \phi(\mathbf{x}_j)>$ is the kernel function, $\mathbf{D}^T\mathbf{D}$ is an arbitrary symmetric decomposition[1] of the matrix with the (i, j)th element being $y_i y_j K(\mathbf{x}_i,\ \mathbf{x}_j)$, and $\lambda \triangleq 1/C$.

The choice of the regularization term λ (or C) can be critical to the performance of SVM training, as it plays an important role in capacity control by maintaining a proper balance between

* Corresponding author.
E-mail address: cqchang@szu.edu.cn (C. Chang).

[1] Actually, the explicit expression of $\mathbf{D}$ is not required, as all computations will be done by using the kernel function $K(\mathbf{x}_i,\ \mathbf{x}_j)$ rather than $\mathbf{D}$.

http://dx.doi.org/10.1016/j.patrec.2017.09.006
0167-8655/© 2017 Elsevier B.V. All rights reserved.

empirical loss and regularization level. Hastie et al. [3] proposed a powerful algorithm (hereafter referred to as the SVMpath), which is capable of fitting the entire path of SVM solutions for every value of the regularization parameter, with essentially the same computational cost as fitting one SVM model. Karasuyama et al. [4] proposed a suboptimal SVMpath algorithm, which allows to control the trade-off between the accuracy and computational cost but brings about a highly degenerate situation of cycling. A new update strategy was further designed to combat the cycling. Note that the path following procedure was also used in other related problems, e.g., support vector regression (SVR) [5], l_1-minimization [6,7] and compressed sensing [8]. As previously reported in the literature, path-following methods that rely on the active set to identify the search direction in each iteration suffer from a lack of provision for singularities [9,10]. It might result in algorithmic failure when, for example, the dataset contains duplicate data points, nearly duplicate points, or points that are linearly dependent in the kernel space.

The singularity problem of SVMpath was recently addressed by Ong et al. in [11], where an improved SVMpath algorithm (termed the ISVMP algorithm) is proposed to handle the singularity problem. It adopted rank-degeneracy testing and additional linear programming to track the optimality condition path in a multidimensional feasible space and derive a search direction strictly satisfying the optimality conditions when a singular active matrix is encountered. However, the procedure of multidimensional feasible space determination is time-consuming. Another shortcoming is that cycling can not be avoided. To solve the problem of algorithmic instability, [11] adopted a procedure of “Backup Routine” (BR) when looping occurs, which will bring more computational complexity.

Astorino and Fuduli [12] proposed a bilevel cross-validation scheme for SVM model selection based on the construction of the entire regularization path. Their methodology of path following, differently from other approaches, works directly on the primal form of SVM, where a general purpose linear programming solver was used to get around the singularity problem. Some heuristics were also proposed to circumvent the singularity problem, e.g., adding a small positive ridge term to the diagonal element of the kernel matrix (called diagonal loading). But this kind of method has at least two major drawbacks: (1) It is very difficult to choose the diagonal adding term suitably in practice. (2) Even though the diagonal adding term is properly chosen, the effect of the added ridge will gradually accumulate along the solution path as more singular events occur.

To avoid the two aforementioned drawbacks, our previous work [13] proposed a new ridge-adding method, which guarantees the existence of the inverse matrix by ensuring that only one index is added into or removed from the active set, rather than directly modifying a singular matrix. Its performance, in terms of both computational complexity and the ability of singularity avoidance, is manifested by rigorous mathematical analyses as well as experimental results. However, the following reasons motivate us to make some changes: (1) The proposed ridge-adding method can only achieve some approximate solutions. (2) Due to the calculation accuracy of computer, adding a very small ridge term might fail to work, especially for RBF kernel. (3) readers sometimes confuse the ridge-adding method with the tradition diagonal loading method.

We notice that all these limitations are caused by the random ridge term added to datasets. Thus, it is more desired to derive a non-ridge-adding-based approach to handle singularities for the SVMpath algorithm. The main novelty of the new proposed method is to divide the active set into the effective and ineffective subsets. The effective subset plays an important role to guarantee the existence of the inverse matrix; while the ineffective subset is to temporarily absorb any index that might cause singularity. Note that the most related work to our proposed method can be found in [14]. Sentelle et al. independently showed that the active matrix $\mathbf{A}_\mathcal{E}$ retains full rank if only one index is added into the active set. They also illustrated that there may exist more than one solution path in the case of degeneracy, and choosing the minimum index in the case of multiple transitions for each event can avoid the cycling. The main difference between this benchmark work and our method is that their method only allows single index to enter into and/or leave out of the active set; while our new method allows multiple indexes to simultaneously enter into and/or leave out of the active set.

2. Review of SVMpath and its singularity

To facilitate our discussion, we briefly review the conventional SVMpath algorithm and its singularity problem in this section.

2.1. The SVMpath algorithm

The SVMpath is piecewise linear by varying of C (or λ) and consequently its construction consists essentially in computing the breakpoints. Following the convention in [11,13], we partition all data points, based on values of the associated α_is, into three subsets, $\mathcal{R}, \mathcal{E}, \mathcal{L}$, which correspond to the index sets containing right, elbow and left points, respectively. Specifically,

$$\begin{aligned} \alpha_i = 0, &\quad \text{if} \quad i \in \mathcal{R} \\ 0 \le \alpha_i \le 1, &\quad \text{if} \quad i \in \mathcal{E} \\ \alpha_i = c_i, &\quad \text{if} \quad i \in \mathcal{L} \end{aligned}$$

where the set $\mathcal{E}$ is also referred to as “active set”. Then, the KKT optimality conditions of the dual form (3) can be written as

$$\mathbf{y}^T \boldsymbol{\alpha} = 0 \tag{4}$$

$$\mathbf{0} \le \boldsymbol{\alpha} \le \mathbf{1} \tag{5}$$

$$\mathbf{d}_i^T \mathbf{D} \boldsymbol{\alpha} - y_i \mu \ge \lambda, \qquad \forall i \in \mathcal{R} \tag{6}$$

$$\mathbf{d}_i^T \mathbf{D} \boldsymbol{\alpha} - y_i \mu = \lambda, \qquad \forall i \in \mathcal{E} \tag{7}$$

$$\mathbf{d}_i^T \mathbf{D} \boldsymbol{\alpha} - y_i \mu \le \lambda, \qquad \forall i \in \mathcal{L} \tag{8}$$

where $\mu \triangleq \lambda b$ and $\mathbf{d}_i$ stands for the ith column of $\mathbf{D}$. According to the property of the KKT optimality conditions, $\boldsymbol{\alpha}$ is an optimal solution to the dual problem (3), if and only if $\boldsymbol{\alpha}$ satisfies all the conditions (4)–(8). We will show how to trace the solution path along the KKT optimality conditions, while evolving λ from a large initialized value to zero.

For notational compactness, we rewrite the KKT conditions (4) and (7) in the form of matrix multiplication:

$$\begin{bmatrix} 0 & -\mathbf{y}^T \\ -\mathbf{y}_\mathcal{E} & \mathbf{D}_\mathcal{E}^T \mathbf{D} \end{bmatrix} \begin{bmatrix} \mu \\ \boldsymbol{\alpha} \end{bmatrix} = \lambda \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \tag{9}$$

where the subscript by an index set for a matrix such as $\mathbf{D}_\mathcal{E}$ indicates a sub-matrix of $\mathbf{D}$ whose columns are indexed by $\mathcal{E}$. Differentiating both sides of (9) with respect to λ yields the following:

$$\underbrace{\begin{bmatrix} 0 & -\mathbf{y}_\mathcal{E}^T \\ -\mathbf{y}_\mathcal{E} & \mathbf{D}_\mathcal{E}^T \mathbf{D}_\mathcal{E} \end{bmatrix}}_{\triangleq \mathbf{A}_\mathcal{E}} \begin{bmatrix} \mu' \\ \boldsymbol{\alpha}'_\mathcal{E} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \tag{10}$$

where $\mu' = \frac{d\mu}{d\lambda}$ and $[\boldsymbol{\alpha}']_i = \frac{d\alpha_i}{d\lambda}$, with $\boldsymbol{\alpha}'_\mathcal{E}$ indicating the sub-vector of $\boldsymbol{\alpha}'$ whose elements are indexed by $\mathcal{E}$. If $\mathbf{A}_\mathcal{E}$ is of *full rank*, we can obtain a search direction corresponding to the movement of $[\begin{smallmatrix}\mu\\ \boldsymbol{\alpha}_{\mathcal{E}}\end{smallmatrix}]$ as follows:

$$\begin{bmatrix}\mu'\\ \boldsymbol{\alpha}'_{\mathcal{E}}\end{bmatrix} = \mathbf{A}_{\mathcal{E}}^{-1}\begin{bmatrix}0\\ \mathbf{1}\end{bmatrix}. \tag{11}$$

To guarantee (9) holding true with the increasing of λ, μ and $\boldsymbol{\alpha}$ are updated as follows:

$$\mu^{(l+1)} = \mu^{(l)} + \Delta\lambda \cdot \mu' \tag{12}$$

$$\boldsymbol{\alpha}_{\mathcal{E}}^{(l+1)} = \boldsymbol{\alpha}_{\mathcal{E}}^{(l)} + \Delta\lambda \cdot \boldsymbol{\alpha}'_{\mathcal{E}} \tag{13}$$

where $(\cdot)^{(l)}$ stands for the value of the corresponding variable at the occurrence of the lth event (referred to the dropping event and adding event in the following), and $\Delta\lambda$ is the stepsize along the direction $[\mu', (\boldsymbol{\alpha}'_{\mathcal{E}})^T]^T$. With the updating equalities (12) and (13), other KKT optimality condition, e.g., (5), (6) or (8), may invalidate with the decreasing of λ. In other words, the solution path might be broken. Either the dropping event or the adding event could lead to such a breakpoint:

- *Dropping event*: An index $j \in \mathcal{E}$ moves into $\mathcal{L} \cup \mathcal{R}$. In other words, the value of α_j attains its lower bound or upper bound. If the lower bound is attained, we have
$$0 = \alpha_j^{(l)} + \Delta\lambda \cdot \alpha'_j \tag{14}$$
while if the upper bound is attained, we have
$$1 = \alpha_j^{(l)} + \Delta\lambda \cdot \alpha'_j. \tag{15}$$
- *Adding event*: An index $j \in \mathcal{L} \cup \mathcal{R}$ moves into $\mathcal{E}$. In this case, either (6) or (8) holds. Hence, we obtain
$$\mathbf{d}_j^T \mathbf{D}\left(\boldsymbol{\alpha}^{(l)} + \Delta\lambda \cdot \boldsymbol{\alpha}'\right) - y_j\left(\mu^{(l)} + \Delta\lambda \cdot \mu'\right) = \lambda^{(l)} + \Delta\lambda. \tag{16}$$

From (14)–(16), the candidates of $\Delta\lambda$, at which one of events happens, can be given as follows:

$$\Delta\lambda_1 = \max_{\Delta\lambda}\left\{\Delta\lambda < 0 : \Delta\lambda = \frac{-\alpha_i^{(l)}}{\alpha'_i}, i \in \mathcal{E}\right\} \tag{17}$$

$$\Delta\lambda_2 = \max_{\Delta\lambda}\left\{\Delta\lambda < 0 : \Delta\lambda = \frac{1-\alpha_i^{(l)}}{\alpha'_i}, i \in \mathcal{E}\right\} \tag{18}$$

$$\Delta\lambda_3 = \max_{\Delta\lambda}\left\{\Delta\lambda < 0 : \Delta\lambda = \frac{\mathbf{d}_i^T\mathbf{D}\boldsymbol{\alpha}^{(l)} - \lambda^{(l)} - y_i\mu^{(l)}}{1 - \mathbf{d}_i^T\mathbf{D}\boldsymbol{\alpha}' + y_i\mu'}, i \in \mathcal{L} \cup \mathcal{R}\right\}. \tag{19}$$

Hence, we can obtain $\lambda^{(l+1)}$ without any event occurrence as

$$\lambda^{(l+1)} = \max\left\{\lambda^{(l)} + \Delta\lambda_1, \lambda^{(l)} + \Delta\lambda_2, \lambda^{(l)} + \Delta\lambda_3\right\}. \tag{20}$$

At the border of the critical region, we update the sets $\mathcal{R}, \mathcal{E}, \mathcal{L}$, and go back to calculate (11) in an iterative way. By this means, the SVMpath algorithm proceeds to track the KKT optimality conditions in the one-dimension space of λ until termination (i.e., $\lambda \to 0$).

2.2. The singularity problem

In order to obtain the search direction in (11), the SVMpath algorithm has to assume that $\mathbf{A}_{\mathcal{E}}$ is of full rank. Actually, almost all path-following methods have to go with this assumption of full rank, e.g., [5,7]. Unfortunately, it is not always valid, especially in the dataset having duplicate data points, or points that are linearly dependent. The presence of such dataset is common among real-world data, and becomes even more prominent in large-scale applications [11]. A violation of this assumption will result in algorithmic instability and thus failure of the SVMpath algorithm.

Our previous work [13] has proved that the sufficient condition for avoiding singularities is that *only one index is added into or removed from the active set at one time*. However, this sufficient condition is not always guaranteed in practical implementations. For example, if the data point, corresponding to the maximum value of $\lambda^{(l+1)}$ in (20), is non-unique, more than one indexes will simultaneously enter into or leave out of the active set. To guarantee the uniqueness of the smallest value of $\Delta\lambda$, our previous work [13] directly introduced a small random ridge term $\Gamma_{i,j}$ into each element of $\mathbf{D}$, which is in fact equivalent to modifying each data point by adding a small random ridge. Due to the randomness of $\Gamma_{i,j}$, the smallest value of $\Delta\lambda$, calculated from (20), is unique almost surely (with probability one). So, only one index corresponding to the smallest value of $\Delta\lambda$ is chosen, and it will be either removed from the active set or added into the active set.

Now, the core question arises: how to directly deal with the problem of the sufficient condition being invalid (i.e., multiple indexes simultaneously enter into and/or leave out of the active set), rather than introducing a random ridge term to each data point? To answer this core question, we try to propose an improved method, which divides the active set into the effective and ineffective subsets. The effective subset plays an important role to guarantee the existence of the inverse matrix; while the ineffective subset is to temporarily absorb any index that might cause singularity. The main difference between the benchmark work [14] and our method is that their method only allows single index to enter into and/or leave out of the active set; while our new method allows multiple indexes to simultaneously enter into and/or leave out of the active set. Due to the multiple active set transitions, the total number of events required by our method is reduced, which will bring an improvement for computational cost (see Section 4).

It is worth noting that allowing multiple indexes to simultaneously enter into and/or leave out of the active set is not new. A framework that traces solution path for many kinds of SVMs was proposed in [15], and they also provided a strategy that can be utilized when the singularities are encountered. The proposed strategy commonly adds or deletes one index at each step; however, it will turn to adding/deleting more than one candidate indexes at one iteration, if every single index always causes a singularity. As the singularity problem is not the main focus of their work, they did not give the explanation why the proposed strategy works. So, it might result in algorithmic instability. Other shortcoming is that they try to search many candidates to form a new non-singular matrix, which will bring a heavy computational cost.

3. Path singularity analysis and the proposed method

In this section, we will give an interesting path singularity analysis for SVMpath, and propose a novel approach to handle the singularity problem. Before providing the main results, we need an auxiliary lemma. We start from introducing Lemma 1 below, which establishes a relationship between the rank of $\mathbf{A}_{\mathcal{E}}$ and the matrix

$$\mathbf{M}_{\mathcal{E}} \triangleq \begin{bmatrix}-\mathbf{y}_{\mathcal{E}}^T\\ \mathbf{D}_{\mathcal{E}}\end{bmatrix}$$

Lemma 1. *The matrix $\mathbf{A}_{\mathcal{E}}$ is of full rank, if and only if the matrix $\mathbf{M}_{\mathcal{E}}$ is column full-rank.*

Proof. See Lemma 1 in [13]. □

3.1. Path singularity analysis

Assume that $\mathbf{A}_{\mathcal{E}}$ is of full rank and the invalidation of the sufficient condition occurs at $\lambda = \lambda^{(l)}$, where P new indexes enter into the active set $\mathcal{E}$ and Q indexes leave out of $\mathcal{E}$ simultaneously. The corresponding adding and removing index sets are denoted as $\mathcal{P}$ and $\mathcal{Q}$. Let $\dot{\mathcal{P}} \subseteq \mathcal{P}$ be a certain set that guarantees $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}}$ being full rank and $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}\cup\{p\}}$ being singular for any $p \in \mathcal{P}\setminus\dot{\mathcal{P}}$. Then, we can prove the following theorem:

Theorem 1. *The KKT optimality conditions hold true with λ changed from $\lambda^{(l)}$ to $\lambda^{(l+1)}$, if we replace the search direction in (11) with*

$$\begin{bmatrix} \mu' \\ \boldsymbol{\alpha}'_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}} \end{bmatrix} = \left(\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}}\right)^{-1} \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \tag{21}$$

Proof. See Appendix. □

Theorem 1 provides new insight to handle singularities. Specifically, if the cardinality of $\mathcal{P}$ (denoted by $|\mathcal{P}|$) is larger than one, the matrix $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\mathcal{P}}$ is possible singular, which may result in a degenerate linear system:

$$\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\mathcal{P}} \begin{bmatrix} \mu' \\ \boldsymbol{\alpha}'_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\mathcal{P}} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}. \tag{22}$$

The degenerate linear system has multiple feasible solutions, but some of them will invalidate the KKT conditions in the next iteration. Theorem 1 demonstrates that adding certain part of elements of $\mathcal{P}$ (i.e., $\dot{\mathcal{P}}$), rather than the whole $\mathcal{P}$, into $\mathcal{E}\setminus\mathcal{Q}$ can guarantee the KKT conditions holding. In this case, the updated active set $\mathcal{E}$ is exactly divided into two parts:

$$\mathcal{E} = \mathcal{E}_E \cup \mathcal{E}_I \tag{23}$$

where $\mathcal{E}_E \triangleq \{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}$ stands for the effective part of the active set that determines the search direction by (21), and $\mathcal{E}_I \triangleq \mathcal{P}\setminus\dot{\mathcal{P}}$ stands for the ineffective part of the active set.

Since the solution provided by Theorem 1 uses partial elements of the active set only, it can be viewed as a degenerate solution for the degenerate linear system (22). Note that there are some similar degenerate solution problems being discussed in the literature. [16] addressed the LASSO problem and uniqueness, where a pseudo-inverse matrix is used to derive the corresponding degenerate path. However, it is rather difficult to extend the corresponding results to SVMpath, because the KKT optimality conditions for SVMpath are much more complex than that of LASSO. [17] showed how to deal with singularities in the active set method for the classical SVM training by generating an infinite direction and illustrated how to update the matrix using Cholesky decomposition. Compared with [17], the solution path provided by Theorem 1 is more efficient, because it allows multiple indexes to simultaneously enter into and/or leave out of the active set. With Theorem 1, the number of iterations required by the proposed method will be greatly reduced.

Finally, it is worthy of noting that $\dot{\mathcal{P}}$ can not be arbitrarily chosen. According to its definition, it must guarantee $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}}$ being full rank and $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}\cup\{p\}}$ being singular for any $p \in \mathcal{P}\setminus\dot{\mathcal{P}}$. But there may exist several feasible $\dot{\mathcal{P}}$s, and there is no need to make $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}}$ be of the maximum rank. Therefore, there are many methods can find such arbitrary set. The most simple one is to do an exhaustive searching as in Algorithm 1.

Algorithm 1 A simple algorithm for finding an appropriate $\dot{\mathcal{P}}$.

```
1. Initialize $\dot{\mathcal{P}} = []$.
2. For $i = 1 : |\mathcal{P}|$, repeat the followings:
   (a) Check the rank of $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}\cup\mathcal{P}(i)}$.
   (b) Update $\dot{\mathcal{P}} = \dot{\mathcal{P}} \cup \mathcal{P}(i)$, if it is full-rank, where $\mathcal{P}(i)$ denotes
       the $i$th index of set $\mathcal{P}$.
3. Output $\dot{\mathcal{P}}$.
```

3.2. The proposed method

Drawn from the above analysis, our proposed method is outlined in Algorithm 2. The algorithm starts with an initialization point provided by some initial configuration methods (Step-1), and then it traces the solution path by recursively computing the breakpoints (Step-2). At each stage, Step-2a is to calculate the search direction with $\mathcal{E}_E$ rather than $\mathcal{E}$ in the original SVMpath; Step-2b is to find the maximum $\lambda^{(l+1)}$ for possible adding or dropping event; Step-2c is to update each variable according to the chosen event; Step-2d is to handle the ineffective subset $\mathcal{E}_I$. Step-2e is to find an appropriate $\dot{\mathcal{P}}$ so as to update $\mathcal{E}_E$ and $\mathcal{E}_I$ according to Theorem 2. The process terminates when λ tends to zero.

Algorithm 2 The proposed solution-path finding algorithm.

```
1. Initialize $\boldsymbol{\alpha}^{(0)}$, $\lambda^{(0)}$, $\mu^{(0)}$, $\mathcal{R}$, $\mathcal{E}$ and $\mathcal{L}$, and let $\mathcal{E}_E = \mathcal{E}$, $\mathcal{E}_I = \mathcal{P} = \mathcal{Q} = []$, and $l = 0$.
2. Repeat the followings:
   (a) Calculate the search direction by solving the KKT system:
       $\begin{bmatrix} 0 & -\mathbf{y}_{\mathcal{E}_E}^T \\ -\mathbf{y}_{\mathcal{E}_E} & \mathbf{D}_{\mathcal{E}_E}^T\mathbf{D}_{\mathcal{E}_E} \end{bmatrix} \cdot \begin{bmatrix} \mu' \\ \boldsymbol{\alpha}'_{\mathcal{E}_E} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}.$
   (b) Find $\lambda^{(l+1)}$ with the maximum step-length:
       $\lambda^{(l+1)} = \max\left\{\lambda^{(l)} + \Delta\lambda_1, \lambda^{(l)} + \Delta\lambda_2, \lambda^{(l)} + \Delta\lambda_3\right\}.$
       where $\Delta\lambda_1$, $\Delta\lambda_2$ and $\Delta\lambda_3$ are similarly defined as in (17)–(19), but $\mathcal{E}$ has to be replaced with $\mathcal{E}_E$.
   (c) Calculate $\mu^{(l+1)}$, $\boldsymbol{\alpha}^{(l+1)}$, and $\mathcal{P}$ and $\mathcal{Q}$ accordingly.
   (d) Update $\mathcal{P} = \mathcal{P} \cup \mathcal{E}_I$.
   (e) Find an appropriate $\dot{\mathcal{P}}$ with Algorithm 1, then update $\mathcal{E}_E$ and $\mathcal{E}_I$ as:
       $\mathcal{E}_E = \{\mathcal{E}_E \setminus \mathcal{Q}\} \cup \dot{\mathcal{P}}, \quad \mathcal{E}_I = \mathcal{P}\setminus\dot{\mathcal{P}}.$
   (f) Let $l = l + 1$, and continue (goto Step-2a) until $\lambda \to 0$.
3. Output the whole solution path.
```

Compared with the original SVMpath algorithm, the main novelty of the new proposed method is to divide the active set into the effective and ineffective subsets. The effective subset plays an important role to guarantee the existence of the inverse matrix in (11), so as to obtain a correct search direction even some singularities are encountered; while the function of ineffective subset is to temporarily absorb any index that might cause singularity, and properly release them into the effective subset when it is necessary. Note that the original SVMpath algorithm can be retrieved as a special case in our framework with the ineffective subset being empty.

Finally, some practical implementation details for the proposed method are given in the following.

- To obtain initializations in Step 1, we can resort to many initial configuration methods, e.g., [3,14,18]. Among them, [14] provided the most efficient method for the initial configuration with imbalanced classes.
- Solving the KKT system in Step-2a could be the most computational expensive step in the proposed method for large-scaled applications. An alternative to solving the KKT system is to use the null-space method with the Cholesky factorization update formula [14,19].
- In Step-2b, numerical overflow might bring some wrong candidates. In the practical implementation, we need to refine the candidates of $\Delta\lambda_1$, $\Delta\lambda_2$ and $\Delta\lambda_3$ to prevent numerical precision errors. Therefore, we introduce some additional filter-conditions into (17)–(19) similarly as [14]. Firstly, we

Table 1
The whole path finding steps with the candidate λs which are obtained from the determination of "dropping event" or "adding event" for linear kernel.

(a) The original SVMpath

lth	$\lambda^{(l)}$	Data point	Event	Cost	Optimal cost
0	7.015	1,5	$\mathcal{L} \to \mathcal{E}$	3.33392730	3.33392730
1	4.125	$1, 5$	$\mathcal{E} \to \mathcal{R}$	2.16666667	2.16666667
		$2, 3, 4$	$\mathcal{L} \to \mathcal{E}$		
–					

(b) Ridge-adding method: $10^{-3} \cdot \texttt{randn}(\text{'seed'}, 0)$

lth	$\lambda^{(l)}$	Data point	Event	Cost	Optimal cost
0	7.01215595	1,5	$\mathcal{L} \to \mathcal{E}$	3.33284597	3.33284637
1	4.25561905	2	$\mathcal{L} \to \mathcal{E}$	2.22432390	2.22432499
2	4.14958737	5	$\mathcal{E} \to \mathcal{R}$	2.17721319	2.17758005
3	4.10827304	4	$\mathcal{L} \to \mathcal{E}$	2.15888257	2.15923246
4	2.06658481	2	$\mathcal{E} \to \mathcal{R}$	1.25166323	1.25181547
5	1.95350131	3	$\mathcal{L} \to \mathcal{E}$	1.20152005	1.20155614
6	1.86992756	4	$\mathcal{E} \to \mathcal{R}$	1.16437921	1.16440613
7	1.56581540	6	$\mathcal{L} \to \mathcal{E}$	1.00208521	1.00211736
8	0	4	$\mathcal{E} \to \mathcal{R}$	0	0
End					

(c) Our method

lth	$\lambda^{(l)}$	Data point	Event	Cost	Optimal cost
0	7.015	1,5	$\mathcal{L} \to \mathcal{E}_E$	3.33392730	3.33392730
1	4.125	$1, 5$	$\mathcal{E}_E \to \mathcal{R}$	2.16666667	2.16666667
		$2, 4$	$\mathcal{L} \to \mathcal{E}_E$		
		3	$\mathcal{L} \to \mathcal{E}_I$		
2	1.875	$2, 4$	$\mathcal{E}_E \to \mathcal{R}$	1.16666667	1.16666667
		3	$\mathcal{E}_I \to \mathcal{E}_E$		
3	1.5625	6	$\mathcal{L} \to \mathcal{E}_E$	1	1
4	0	3,6	$\mathcal{E}_E \to \mathcal{R}$	0	0
End					

add $\alpha_i' > 0$ into (17), because of $-\alpha_i^{(l)}/\alpha_i' < 0$ and $\alpha_i^{(l)} \geq 0$. Secondly, we add $\alpha_i' < 0$ into (18), because of $1 - \alpha_i^{(l)}/\alpha_i' < 0$ and $\alpha_i^{(l)} \leq 1$. Thirdly, we add $1 - \mathbf{d}_i^T \mathbf{D}\boldsymbol{\alpha}' + y_i\mu' < 0, i \in \mathcal{L}$ and $1 - \mathbf{d}_i^T \mathbf{D}\boldsymbol{\alpha}' + y_i\mu' > 0, i \in \mathcal{R}$ into (19) because of (6) and (8).

- From the proof of Theorem 1, each index $j \in \mathcal{E}_I$ remains on the margin, i.e., $\mathbf{d}_j^T \mathbf{D}\boldsymbol{\alpha} - y_j\mu = \lambda$. Hence, if the ineffective subset $\mathcal{E}_I$ of the previous step is nonempty, we have to absorb all the elements into the current set $\mathcal{P}$ (see Step-2d).
- To find $\dot{\mathcal{P}}$ in Step-2e, we have to check the ranks of $\mathbf{A}_{\{\mathcal{E}_E \backslash \mathcal{Q}\} \cup \dot{\mathcal{P}} \cup \mathcal{P}(i)}, i = 1, 2, \ldots, |\mathcal{P}|$. Rank calculation involves a massive computation, and is also not robust to numerical precision errors. To solve this problem, we use the Cholesky factorization update formula, and then we only need to check the new added diagonal element. Since the former Cholesky factorization has been calculated in Step-2a, the computational cost of rank calculations will be reduced greatly. Note that the tolerance chosen in this step for detecting the singularity is 10^{-5}.

With these implementations, the computational complexities associated with Steps-2a and 2b in each iteration are $\mathcal{O}(N_E^2)$ and $\mathcal{O}(NN_E)$, receptively, where N_E stands for the average of $|\mathcal{E}_E|$. The complexity required by each loop in Algorithm 1 is $\mathcal{O}(N_E^2)$. Therefore, this suggests the total computational requirement of the proposed method is $\mathcal{O}(LNN_E + (L+M)N_E^2)$, where M is the total number of loops through Algorithm 1. It is very similar to SVMpath and SSVMP, whose computational complexities are $\mathcal{O}(LNN_E + LN_E^3)$ and $\mathcal{O}(LNN_E + LN_E^2)$, respectively. However, it is worth noting that: (1) the total number of events required by our method is less than others; (2) the total number of loops through Algorithm 1 is usually much smaller than the total number of events. Hence, the proposed method has a computational advantage over others. For example, in the first simulation of Section 4 (see Table 1), the total number of events required by our method is half of others; while the total number of loops through Algorithm 1 is only three.

4. Simulation results

In this section, we will run some simulations to illustrate the correctness and performance of our proposed method, where two commonly used kernels are used: (1) linear kernel where $K_{i,j} = \mathbf{x}_i^T \mathbf{x}_j$; (2) RBF kernel where $K_{i,j} = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|_2^2)$. The computer codes are developed in MATLAB 8.3.0.532 and run on an ADM A10-7890K 4.10 GHz with 16 GB RAM under the Windows 10 operation system.

To verify the validity of our method and illustrate that it works well under singular case, a toy example is first in order. Consider a 2-D dataset in the form of (i: y_i, x_i):

$$\{(1:1, (0.7, 0.3)), (2:1, (0.5, 0.5)), (3:1, (0.5, 0.5)),$$
$$(4:-1, (2.0, 2.0)), (5:-1, (1.4, 2.6)), (6:-1, (1.75, 1.75))\}.$$

Note that points 2 and 3 are duplicates of each other, and points 1, 2 (or 3), 4 and 5 are linearly dependent. Other examples can be found in [11], where the datasets with linearly dependent points and duplicate points are considered, respectively. Table 1 shows the whole path finding steps with the candidate λs, obtained from the determination of dropping event or adding event for linear kernel. As shown in Table 1-a), the original SVMpath is disturbed when calculating the new search direction at $\lambda = 4.125$, as the active set contains duplicate points 2 and 3. The values of $\lambda^{(l)}$ in Table 1-b) for the ridge-adding method are slightly different from Table 1-a) due to the added ridges. This tiny modification ensures that only one data point moves from one set into another with the change of λ, and the singularities are successfully avoided at $\lambda = 4.125$ and 1.875. However, the ridge-adding method has a little performance loss caused by the added ridges.[2] As shown in Table 1-c), our proposed method divides the active set into

[2] Note that the value of "Cost" is evaluated by $-\frac{1}{2\lambda^{(l)}}(\boldsymbol{\alpha}^{(l)})^T \mathbf{D}^T \mathbf{D}\boldsymbol{\alpha}^{(l)} + \mathbf{1}^T \boldsymbol{\alpha}^{(l)}$ directly; while the value of "Optimal cost" is evaluated by solving the problem (3) with the same $\lambda^{(l)}$ via CVX.

Table 2
Results achieved by ISVMP, SSVMP and our method for linear kernel.

Dataset	ISVMP				SSVMP				Our method					
	l_{max}	$\|\mathcal{E}\|_{max}$	N_{BR}	T (s)	l_{max}	$\|\mathcal{E}\|_{max}$	N_{RP}	T (s)	l_{max}	$\|\mathcal{E}_E\|_{max}$	$\|\mathcal{E}_I\|_{max}$	N_{A1}	N_{RP}	T (s)
Australian	1497	15	0	35.89	1596	15	82	4.723	1521	15	4	28	45	2.428
Banknote	3677	5	0	91.23	3918	5	240	23.59	3639	5	6	51	94	11.69
Diabetes	462	9	0	10.05	477	9	14	3.294	452	9	3	7	2	1.003
German	415	29	3	73.78	452	25	20	6.476	439	25	2	6	14	1.575
Heart	473	15	0	5.036	501	14	25	0.491	468	14	4	7	9	0.247
Hillvalley	473	17	0	12.23	498	17	22	1.173	467	17	3	6	7	0.674
Ionosphere	708	34	0	5.906	743	34	33	1.531	692	34	4	28	8	0.684
svmguide1	5103	5	0	297.7	5551	5	447	165.5	5056	5	4	72	167	68.32
wbc	1160	28	0	12.56	1228	28	64	3.408	1145	28	3	14	20	1.363

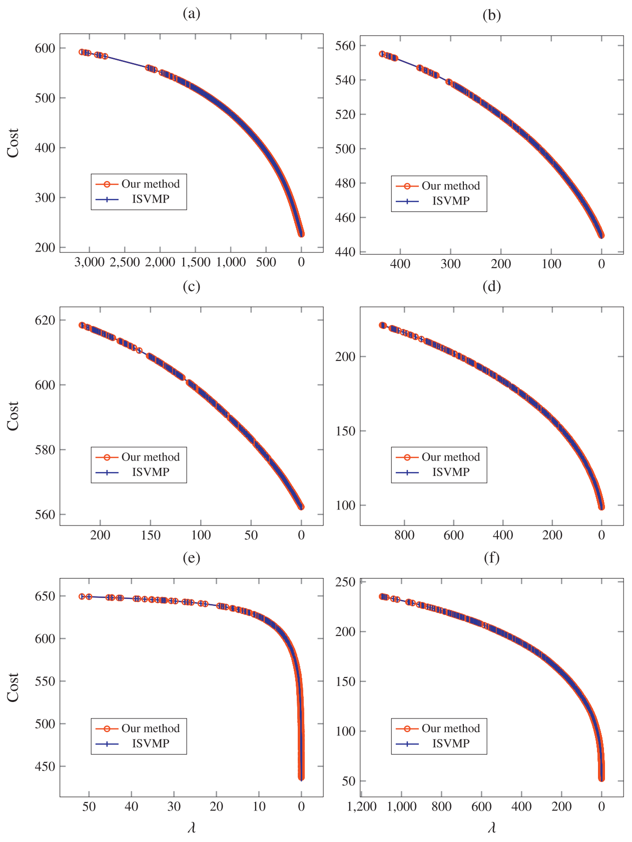

Fig. 1. The cost value comparisons between ISVMP and our method for linear kernel. (a) Australian; (b) Diabetes; (c) German; (d) Heart; (e) Hillvalley; (f) Ionosphere.

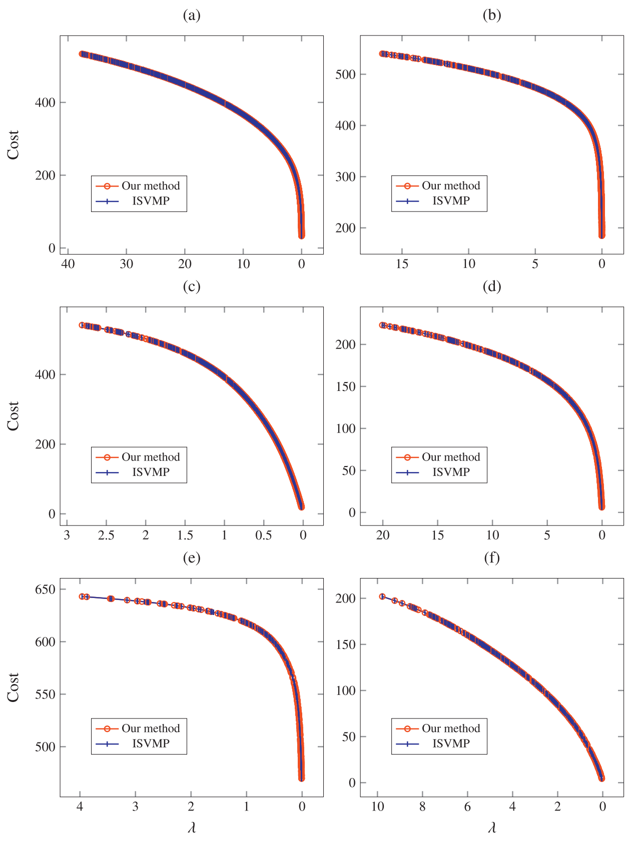

Fig. 2. The cost value comparisons between ISVMP and our method for RBF kernel with $\gamma = 0.1$. (a) Australian; (b) Diabetes; (c) German; (d) Heart; (e) Hillvalley; (f) Ionosphere.

the effective and ineffective subsets, where the ineffective subset temporarily absorbs the index 3 that causes singularity at $\lambda = 4.125$. Compared with the ridge-adding method, there is no any performance loss in the proposed method.

In the next simulation, we use nine real datasets, obtained from the University of California at Irvine (UCI) repository [20] and the Library for Support Vector Machines website [21], to testify the performance under practical datasets. As the ridge-adding method only achieves an approximate solution and sometimes it might fail to work, in the following, we only compare our method with ISVMP [11] and SSVMP [14]. In order to enhance the singularity probability, duplicate data points are added by randomly sampling 10 percent of the data points in each dataset. Then, each feature vector is normalized to have zero mean and unit variance. Usually, the classes in each dataset are imbalanced. If the initializations are required, we will resort to the initial configuration method proposed in [14]. Note that the original SVMpath will always fails to work with such duplicate datasets. Tables 2 and 3 show the results achieved by the three methods for linear kernel and RBF kernel, respectively. The quantities $|\mathcal{E}|_{max}$, $|\mathcal{E}_E|_{max}$, $|\mathcal{E}_I|_{max}$ refer to the maximal cardinalities of the sets $\mathcal{E}$, $\mathcal{E}_E$ and $\mathcal{E}_I$, respectively. The quantities l_{max}, N_{BR}, N_{RP}, N_{A1} and T refer to the total number of events among the path, the number of times the backup routine (BR) is invoked due to KKT violation or cycling for the ISVMP algorithm, the number of repeat λ events, the number of times Algorithm 1 is invoked for the proposed method, and the runtime used for each method, respectively. Figs. 1 and 2 show the cost value comparisons between our method and ISVMP for linear kernel and RBF kernel, respectively. Several observations are obtained from these results: (1) Two cost curves in the figures coincide with each other. Since ISVMP will invoke a backup routine to guarantee reaching the optimality at each breakpoint, such coincidence implies that our method also achieves the optimal solution at each

Table 3
Results achieved by ISVMP, SSVMP and our method for RBF kernel with $\gamma = 0.1$.

Dataset	ISVMP				SSVMP				Our method					
	l_{max}	$\lvert\mathcal{E}\rvert_{max}$	N_{BR}	T (s)	l_{max}	$\lvert\mathcal{E}\rvert_{max}$	N_{RP}	T (s)	l_{max}	$\lvert\mathcal{E}_E\rvert_{max}$	$\lvert\mathcal{E}_I\rvert_{max}$	N_{A1}	N_{RP}	T (s)
Australian	1258	301	9	300.5	1368	272	58	8.686	1257	272	14	81	3	6.915
Banknote	3414	20	0	83.88	3601	20	186	22.36	3304	20	12	103	38	10.45
Diabetes	942	246	0	24.78	989	246	39	8.331	912	246	13	13	1	5.379
German	751	835	8	2136	830	756	42	66.15	775	756	15	29	15	62.97
Heart	451	148	0	5.448	478	147	25	1.103	427	147	8	31	0	0.858
Hillvalley	319	104	10	63.53	360	93	25	2.349	331	93	5	25	11	1.957
Ionosphere	369	205	7	102.4	423	183	26	1.896	372	183	12	26	1	1.282
svmguide1	5059	37	0	303.3	5377	37	314	165.0	4854	37	5	209	51	67.67
wbc	711	226	0	71.24	790	209	39	4.000	717	209	8	43	3	2.428

breakpoint. (2) The singularity problem is frequently encountered in each dataset due to the fact that they contains duplicate points and points that are linearly dependent in the kernel space. Nevertheless, our method can always properly handle the singularity problem. (3) Our method is faster than ISVMP and SSVMP, and the total number of events required by our method is less than that of ISVMP and SSVMP.

5. Conclusion

This paper proposes a non-ridge-adding-based approach to fill the gap in handling the singularity problem of the SVMpath algorithm. The main novelty of the new method is to divide the active set $\mathcal{E}$ into $\mathcal{E}_E$ and $\mathcal{E}_I$, so as to obtain a correct search direction even some singularities are encountered. One limitation of our method is that it requires an additional searching for a proper set $\dot{\mathcal{P}}$ (Algorithm 1). Therefore, its computational complex is higher than the original SVMpath or the ridge-adding method. For future work, it will be interesting to see if one can derive some efficient searching algorithms for $\dot{\mathcal{P}}$.

Acknowledgments

J. Dai's work is supported by the National Natural Science Foundation of China (NSFC) under Project 61571211 and the Jiangsu Oversea Research & Training Program for University Prominent Young & Middle-aged Teachers and Presidents. W. Xu's work is supported by the NSFC under Project 61271380 and the Guangdong Natural Science Foundation under Project 2014A030313515. C. Chang's work is supported by Shenzhen Talent Peacock Plan (827-000083), Shenzhen Peacock Team Plan KQTD2015033016104926), Guangdong Pearl River Talents Plan Innovative and Entrepreneurial Team grant (2016ZT06S220).

Appendix

Since $\lambda^{(l)}$ and $\lambda^{(l+1)}$ refer to values of the regularization parameter λ at the lth and $(l+1)$th breakpoints, respectively, neither the dropping event nor the adding event occurs between $\lambda^{(l)}$ and $\lambda^{(l+1)}$. Hence, to prove the KKT optimality conditions hold true, we just have to show that (4) and (7) hold true when λ is changed from $\lambda^{(l)}$ to $\lambda^{(l+1)}$.

The whole set $\{\mathcal{E} \setminus \mathcal{Q}\} \cup \mathcal{P}$ can be divided into two subsets: $\{\mathcal{E} \setminus \mathcal{Q}\} \cup \dot{\mathcal{P}}$ and $\mathcal{P} \setminus \dot{\mathcal{P}}$. From (9) and (10), it is easy to verify that

$$\begin{bmatrix} 0 & -\mathbf{y}^T \\ -\mathbf{y}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}} & \mathbf{D}^T_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}}\mathbf{D} \end{bmatrix} \begin{bmatrix} \mu^{(l)} + \Delta\lambda \cdot \mu' \\ \boldsymbol{\alpha}^{(l)} + \Delta\lambda \cdot \boldsymbol{\alpha}' \end{bmatrix} = (\lambda^{(l)} + \Delta\lambda) \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \tag{24}$$

where the active elements in $\boldsymbol{\alpha}'$ (denoted by $\boldsymbol{\alpha}'_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}}$) and μ' are calculated by (11) with $\mathbf{A}_{\mathcal{E}} = \mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}}$. It is equivalent to rewrite (24) as

$$-\mathbf{y}^T(\boldsymbol{\alpha}^{(l)} + \Delta\lambda \cdot \boldsymbol{\alpha}') = 0 \tag{25}$$

and

$$\mathbf{d}_p^T(\boldsymbol{\alpha}^{(l)} + \Delta\lambda \cdot \boldsymbol{\alpha}') - y_p(\mu^{(l)} + \Delta\lambda \cdot \mu') \\ = (\lambda^{(l)} + \Delta\lambda), \quad \forall p \in \{\mathcal{E} \setminus \mathcal{Q}\} \cup \dot{\mathcal{P}}. \tag{26}$$

To finish the proof, what remains is to show (26) also holds true for any $p \in \mathcal{P} \setminus \dot{\mathcal{P}}$, if $\mathcal{P} \setminus \dot{\mathcal{P}}$ is nonempty. As $\mathbf{A}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}\cup\{p\}}$ is singular for any $p \in \mathcal{P} \setminus \dot{\mathcal{P}}$, there always exists a non-zero vector $\boldsymbol{\beta}_p$ for any $p \in \mathcal{P} \setminus \dot{\mathcal{P}}$ such that (Lemma 1):

$$\begin{bmatrix} -\mathbf{y}^T_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}} \\ \mathbf{D}_{\{\mathcal{E}\setminus\mathcal{Q}\}\cup\dot{\mathcal{P}}} \end{bmatrix} \boldsymbol{\beta}_p = \begin{bmatrix} -y_p \\ \mathbf{d}_p \end{bmatrix}. \tag{27}$$

Multiplying both sides of (24) left by $\begin{bmatrix} 0 & \boldsymbol{\beta}_p^T \end{bmatrix}$ yields

$$\mathbf{d}_p^T(\boldsymbol{\alpha}^{(l)} + \Delta\lambda \cdot \boldsymbol{\alpha}') - y_p(\mu^{(l)} + \Delta\lambda \cdot \mu') \\ = (\lambda^{(l)} + \Delta\lambda) \cdot \boldsymbol{\beta}_p^T \mathbf{1}, \quad \forall p \in \mathcal{P} \setminus \dot{\mathcal{P}}. \tag{28}$$

As the index $p \in \mathcal{P} \setminus \dot{\mathcal{P}}$ enters into the active set at $\lambda = \lambda^{(l)}$, we have

$$\lambda^{(l)} = \mathbf{d}_p^T \boldsymbol{\alpha}^{(l)} - y_p \mu^{(l)} = \lambda^{(l)} \cdot \boldsymbol{\beta}_p^T \mathbf{1} \tag{29}$$

where (29) follows from (28) with $\Delta\lambda = 0$. With (28) and $\boldsymbol{\beta}_p^T \mathbf{1} = 1$, we obtain

$$\mathbf{d}_p^T(\boldsymbol{\alpha}^{(l)} + \Delta\lambda \cdot \boldsymbol{\alpha}') - y_p(\mu^{(l)} + \Delta\lambda \cdot \mu') = (\lambda^{(l)} + \Delta\lambda), \forall p \in \mathcal{P} \setminus \dot{\mathcal{P}}.$$

References

[1] F. Simistira, V. Katsouros, G. Carayannis, Recognition of online handwritten mathematical formulas using probabilistic svms and stochastic context free grammars, Pattern Recognit. Lett. 53 (2015) 85–92.
[2] A. Singla, S. Patra, L. Bruzzone, A novel classification technique based on progressive transductive svm learning, Pattern Recognit. Lett. 42 (2014) 101–106.
[3] T. Hastie, S. Rosset, R. Tibshirani, J. Zhu, The entire regularization path for the support vector machine, J. Mach. Learn. Res. 5 (2004) 1391–1415.
[4] M. Karasuyama, I. Takeuchi, Suboptimal solution path algorithm for support vector machine, ICML 2011 (2011).
[5] G. Wang, D. Yeung, F.H. Lochovsky, A new solution path algorithm in support vector machines, IEEE Trans. Neural Netw. 19 (2008) 1753–1767.
[6] S. Rosset, J. Zhu, Piecewise linear regularized solution paths, Ann. Stat. 35 (2007) 1012–1030.
[7] M.S. Asif, J. Romberg, Sparse recovery of streaming signals using l_1-homotopy, IEEE Trans. Signal Process. 62 (16) (2014) 4209–4223.
[8] D.L. Donoho, Y. Tsaig, Fast solution of l_1-norm minimization problems when the solution may be sparse, preprint, (2006). http://www.stanford.edu/tsaig/research.html.
[9] C.G. Sentelle, G.C. Anagnostopoulos, M. Georgiopoulos, Efficient revised simplex method for SVM training, IEEE Trans. Neural Netw. 22 (2011) 1650–1661.
[10] A. Shilton, M. Palaniswami, D. Ralph, A.C. Tsoi, Incremental training of support vector machines, IEEE Trans. Neural Netw. 16 (2005) 114–131.
[11] C. Ong, S. Shao, J. Yang, An improved algorithm for the solution of the regularization path of support vector machine, IEEE Trans. Neural Netw. 21 (2010) 451–462.
[12] A. Astorino, A. Fuduli, The proximal trajectory algorithm in svm cross validation, IEEE Trans. Neural Netw. Learn. Syst. 27 (5) (2016) 966–977.
[13] J. Dai, C. Chang, F. Mai, D. Zhao, W. Xu, On the svmpath singularity, IEEE Trans. Neural Netw. Learn. Syst. 24 (11) (2013) 1736–1748.

[14] C.G. Sentelle, G.C. Anagnostopoulos, M. Georgiopoulos, A simple method for solving the svm regularization path for semidefinite kernels, IEEE Trans. Neural Netw. Learn. Syst. 27 (4) (2016) 709–722.
[15] Z.-l. Wu, A. Zhang, C.-h. Li, A. Sudjianto, Trace solution paths for svms via parametric quadratic programming, KDD Worskshop: Data Mining Using Matrices and Tensors (2008).
[16] R.J. Tibshirani, The lasso problem and uniqueness, Electron. J. Stat. 7 (2013) 1456–1490.
[17] K. Scheinberg, An efficient implementation of an active set method for svms, J. Mach. Learn. Res. 7 (Oct) (2006) 2237–2257.
[18] J. Dai, F. Mai, On SVMpath initialization, Signal Process. 92 (2012) 1258–1267.
[19] J. Norcedal, S.J. Wright, Springer series in operations research and financial engineering, Numerical Optimization, 1999.
[20] A. Frank, A. Asuncion, UCI Machine Learning Repository [Online], 2010. Available: http://archive.ics.uci.edu/ml.
[21] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27.