Nonlinear Regularization Path for Quadratic Loss Support Vector Machines

Masayuki Karasuyama, *Student Member, IEEE,* Ichiro Takeuchi, *Member, IEEE,*

***Abstract*—Regularization path algorithms have been proposed to deal with model selection problem in several machine learning approaches. These algorithms allow to compute the entire path of solutions for every value of regularization parameter using the fact that their solution paths have piecewise linear form. In this paper we extend the applicability of regularization path algorithm to a class of learning machines that have quadratic loss and quadratic penalty term. This class contains several important learning machines such as squared hinge loss SVM and modified Huber loss SVM. We first show that the solution paths of this class of learning machines have piecewise nonlinear form, and piecewise segments between two breakpoints are characterized by a class of rational functions. Then we develop an algorithm that can efficiently follow the piecewise nonlinear path by solving these rational equations. To solve these rational equations, we use rational approximation technique with quadratic convergence rate, and thus, our algorithm can follow the nonlinear path much more precisely than existing approaches such as predictor-corrector type nonlinear-path approximation. We show the algorithm performance on some artificial and real data sets.**

***Index Terms*—support vector machines, parametric programming, rational approximation**

I. Introduction

In this paper we study binary classification problem with training data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ is the input and $y_i \in \{1, -1\}$ is the output class label. We consider a linear discriminant function:

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \Phi(\boldsymbol{x}),$$

where Φ maps the data into some feature space $\mathcal{F}$. A large class of learning machines are formulated as minimization of the following regularized risk function:

$$\min_{\boldsymbol{w}} \quad \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i)) + \lambda J(\boldsymbol{w}), \tag{1}$$

where ℓ is a loss function, J is a penalty function, and $\lambda > 0$ is a regularization parameter. In order to obtain a discriminant function with good generalization property, we need to select a good λ that appropriately controls the model complexity (model selection). A typical model selection approach is to specify a list of candidate values of λ and then apply cross validation to select the best one. However, this procedure is time-consuming since we need to solve many optimization problems in various settings.

Regularization path algorithm [21] provides an efficient approach to model selection problem. It allows to compute the entire solution path for a range of regularization parameters λ. The regularization path algorithm is built on an optimization technique called *parametric programming* also a.k.a. *path-following* [1], [16]. Recently, in the machine learning literature, path-following was used for various purposes (e.g. [3], [9], [19], [22], [25], [26], [28], [35], [41], [29]). Most regularization path algorithms in the literature were developed by exploiting the fact that the solution paths in a class of learning machines have *piecewise linear* form. For example, the support vector machine (SVM) [37] characterized by the following hinge loss:

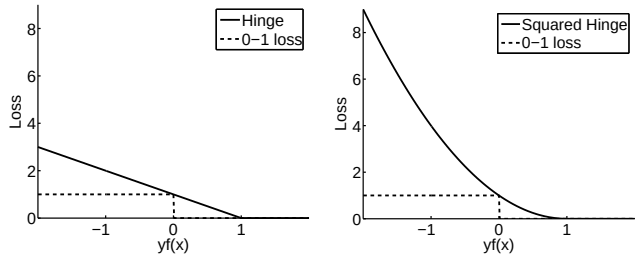
$$\ell(y_i, f(\boldsymbol{x}_i)) = \max(0, 1 - y_i f(\boldsymbol{x}_i)) \tag{2}$$

and ℓ_2-norm penalty term: $J(\boldsymbol{w}) = \frac{1}{2}\|w\|_2^2$ is shown to have piecewise linear regularization path [21]. Recently, [34] showed that the regularization path has piecewise linear form if the loss function ℓ is a piecewise quadratic function and the penalty term $J(\boldsymbol{w})$ is a piecewise linear function. The LASSO [36] is a typical example of this class of learning machine. In optimization literature, [32] has derived more general sufficient conditions for piecewise linearity in quadratic and linear programming problems.

In other class of learning machines, the solution path may have nonlinear form. *Predictor corrector* approach [1] is usually adopted for general nonlinear path-following. In predictor corrector approach, the predictor step and the corrector step are iterated: the predictor step approximates the nonlinear (curved) solution path (in many cases, using Taylor expansion), while the corrector step projects the predicted solution to the solution space so that it satisfies the optimality conditions. This approach have been applied to some learning problems [4], [24], [30]. Some other approaches are also proposed in machine learning literature. For example, [33] proposed second-order approximation algorithm for nonlinear regularization path, in which small step is taken and the approximation is updated at each iteration. As another example, [40] derived an updating formula to obtain the path of solutions along the change of the kernel parameter (such as standard deviation in Gaussian kernel). In these methods, we can only obtain roughly approximated nonlinear path of solutions. If we want these nonlinear approximated solution path to be accurate, the algorithm would be computationally demanding because we need to take very small steps.

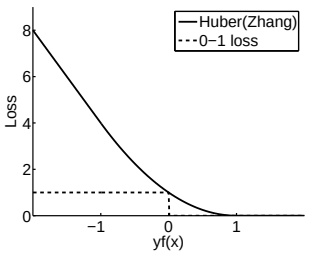
In this study we consider a class of learning machines that have quadratic loss and quadratic penalty. This class contains several important learning machines such as *squared hinge loss SVM* and *modified Huber loss SVM*. These loss functions are formulated as

The authors with the Department of Engineering, Nagoya Institute of Technology, Nagoya, Aichi, 466-8555 Japan e-mail: (krsym@goat.ics.nitech.ac.jp, and takeuchi.ichiro@nitech.ac.jp).

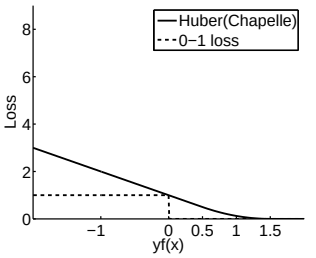


(a) Hinge loss (2)

(b) Squared hinge loss (3)



(c) Modified Huber loss (4)



(d) Modified Huber loss (5) ($\varepsilon = 0.5$)

Fig. 1. Loss functions

- Squared hinge loss:

$$\ell(y, f(\boldsymbol{x})) = \max(0, 1 - yf(\boldsymbol{x}))^2. \tag{3}$$

- Modified Huber loss [43]:

$$\ell(y, f(\boldsymbol{x})) = \begin{cases} 0, & yf(\boldsymbol{x}) > 1, \\ (1 - yf(\boldsymbol{x}))^2, & yf(\boldsymbol{x}) \in [-1, 1], \\ -4yf(\boldsymbol{x}), & yf(\boldsymbol{x}) < -1. \end{cases} \tag{4}$$

Another formulation of Huber-type loss function proposed in [11]:

$$\ell(y, f(\boldsymbol{x})) = \begin{cases} 0, & yf(\boldsymbol{x}) > 1 + \varepsilon, \\ \frac{(1+\varepsilon - yf(\boldsymbol{x}))^2}{4\varepsilon}, & |1 - yf(\boldsymbol{x})| \le \varepsilon, \\ 1 - yf(\boldsymbol{x}), & yf(\boldsymbol{x}) < 1 - \varepsilon, \end{cases} \tag{5}$$

where $\varepsilon > 0$ is a parameter. If $\varepsilon \to 0$, this loss function approaches to the hinge loss.

Fig. 1 shows these loss functions along with the 0-1 loss. These quadratic loss functions are sometimes preferred to the hinge loss. For example, it is known that this type of loss functions are suited to estimate conditional probability $P(Y = 1 \mid X = \boldsymbol{x})$ (see e.g., [5], [43]). Another advantage of these loss functions is their differentiability. Some primal SVM solvers [6], [11], [23] require differentiable objective function.

Unfortunately, the regularization paths of this class of learning machines (quadratic loss + quadratic penalty) do not exhibit piecewise linear form anymore. To extend the applicability of regularization path algorithm, we develop a nonlinear regularization path algorithm for this class of learning machines. We first show that the solution path of this class of learning machine is represented as piecewise nonlinear form, and the piecewise segment of solutions between two breakpoints are characterized by a class of rational function. The breakpoint itself can be identified solving the rational equations. Then we develop an efficient algorithm that can efficiently follow the piecewise nonlinear path by solving these rational equations. To solve these rational equations, we introduce a rational approximation technique with quadratic convergence rate used in rank-one-update of eigenvalue decompositions [8]. Note that our algorithm is NOT a predictor-corrector type approach. While predictor corrector approach can only follow nonlinear path with rather rough approximation, our algorithm can compute accurate path of solutions because we use an efficient iterative procedure with quadratic convergence rate. Fig. 2 illustrates the differences among our approach and the other nonlinear path following strategies.

The rest of the paper is organized as follows. Section II formulates learning machines with quadratic loss function and quadratic penalty. We restrict the penalty term to have the form: $J(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$, and consider a class of quadratic loss functions, this type of learning machines are sometimes referred to as *quadratic loss support vector machine (SVM)*. In Section III, we describe our nonlinear regularization path algorithm for quadratic loss SVM. After presenting experimental results in Section IV, we close in Section V with concluding remarks.

II. The Support Vector Machines with a Quadratic Loss Function

In this paper, we set the penalty term as $J(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2$ and the loss function as the following general quadratic loss function:

$$\ell(y, f(\boldsymbol{x})) = \begin{cases} 0, & yf(\boldsymbol{x}) > \rho, \\ (\rho - yf(\boldsymbol{x}))^2, & yf(\boldsymbol{x}) \in [\rho - h, \rho], \\ 2h(\rho - yf(\boldsymbol{x})) - h^2, & yf(\boldsymbol{x}) < \rho - h, \end{cases} \tag{6}$$

where $\rho > 0$ and $h > 0$. The loss function (6) can represent the previous three quadratic loss functions (3)-(5) by specifying (ρ, h). If we set $(\rho, h) = (1, \infty), (1, 2)$ and $(1 + \varepsilon, 2\varepsilon)$, the loss function (6) is reduced to (3),(4) and (5), respectively[1]. The optimization problem (1) is now written as

$$\begin{aligned} \min_{\boldsymbol{w}, \{\xi_i\}_{i=1}^n} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{2}\sum_{i=1}^n \phi(\xi_i), \\ \text{s.t.} \quad & \rho - y_i f(\boldsymbol{x}_i) \le \xi_i,\ \xi_i \ge 0, i = 1, \cdots, n, \end{aligned}$$

where

$$\phi(\xi_i) = \begin{cases} \xi_i^2, & \xi_i \in [0, h], \\ 2h\xi_i - h^2, & \xi_i > h. \end{cases}$$

We derive the dual problem using the same approach in [12]. Introducing Lagrange multipliers $\alpha_i, \eta_i \ge 0$, $i = 1, \cdots, n$, we can write the corresponding Lagrangian as

$$\begin{aligned} L = \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{2}\sum_{i=1}^n \phi(\xi_i) + \\ \sum_{i=1}^n \alpha_i\{\rho - y_i \boldsymbol{w}^\top \Phi(\boldsymbol{x}_i) - \xi_i\} - \sum_{i=1}^n \eta_i \xi_i. \end{aligned} \tag{7}$$

[1]To represent (5) by (6), we further need to multiply (6) by $1/(4\varepsilon)$. This difference can be absorbed by the scale of regularization parameter.

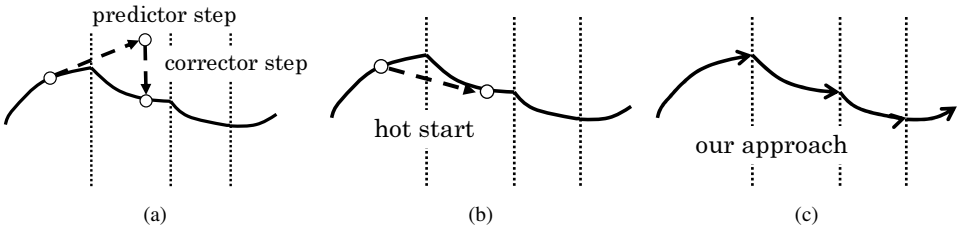


Fig. 2. Schematic illustrations of the differences among several nonlinear path following approaches. In each illustration, the piecewise thick curves represent the nonlinear solution path and vertical dashed lines indicate the breakpoints. (a) Predictor corrector approach iterates the predictor step and the corrector step. The predictor step approximates the solution along the path and the corrector step brings the predicted point back to the path. (b) Hot start approach uses previous solution for initial estimation of the next solution. (c) In our approach the analytical form of the nonlinear solution path is derived and the breakpoints can be detected exactly. The first two approaches (a) and (b) roughly approximate the nonlinear path and they could not detect the breakpoints.

Setting the derivatives w.r.t. primal variables $\boldsymbol{w}$ and ξ_i to zero, we obtain

$$\frac{\partial L}{\partial \boldsymbol{w}} = \mathbf{0} \quad \Leftrightarrow \quad \boldsymbol{w} = \frac{1}{\lambda}\sum_{i=1}^{n} \alpha_i y_i \Phi(\boldsymbol{x}_i), \tag{8}$$

$$\begin{aligned} \frac{\partial L}{\partial \xi_i} = 0 \quad &\Leftrightarrow \quad \frac{1}{2}\frac{\partial \phi(\xi_i)}{\partial \xi_i} = \alpha_i + \eta_i, \\ &\Leftrightarrow \quad \begin{cases} \xi_i &= \alpha_i + \eta_i, \quad \xi_i \in [0, h], \\ h &= \alpha_i + \eta_i, \quad \xi_i > h. \end{cases} \end{aligned} \tag{9}$$

Substituting (8) into (7), we obtain

$$\begin{aligned} L \;=\; & -\frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j Q_{ij} + \rho \sum_{i=1}^{n} \alpha_i \\ & + \sum_{i=1}^{n} \{\frac{1}{2}\phi(\xi_i) - (\alpha_i + \eta_i)\xi_i\}, \end{aligned}$$

where $Q_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Using (9), we can eliminate ξ_i from Lagrangian. If $\xi_i \in [0, h]$, we have

$$\begin{aligned} \frac{1}{2}\phi(\xi_i) - (\alpha_i + \eta_i)\xi_i \;&=\; \frac{1}{2}\xi_i^2 - (\alpha_i + \eta_i)\xi_i \\ &=\; -\frac{1}{2}(\alpha_i + \eta_i)^2. \end{aligned}$$

On the other hand, if $\xi_i > h$,

$$\begin{aligned} \frac{1}{2}\phi(\xi_i) - (\alpha_i + \eta_i)\xi_i \;&=\; h\xi_i - \frac{1}{2}h^2 - h\xi_i \\ &=\; -\frac{1}{2}(\alpha_i + \eta_i)^2. \end{aligned}$$

Then, the dual problem is represented as

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\eta}} \;\; & W(\boldsymbol{\alpha}, \boldsymbol{\eta}) = -\frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j Q_{ij} \\ & \qquad + \rho \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n} (\alpha_i + \eta_i)^2, \\ \text{s.t.} \;\; & \alpha_i, \eta_i \geq 0, \;\; \alpha_i + \eta_i \leq h, \;\; i = 1, \cdots, n, \end{aligned}$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_n]^\top$ and $\boldsymbol{\eta} = [\eta_1, \cdots, \eta_n]^\top$. Since $\alpha_i, \eta_i \geq 0$, an inequality $W(\boldsymbol{\alpha}, \boldsymbol{\eta}) \leq W(\boldsymbol{\alpha}, \mathbf{0})$ holds for every feasible $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$. Therefore, we can delete $\boldsymbol{\eta}$ and the dual problem is finally written as

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \;\; & W(\boldsymbol{\alpha}) = -\frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j Q_{ij} \\ & \qquad + \rho \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n} \alpha_i^2, \\ \text{s.t.} \;\; & 0 \leq \alpha_i \leq h, \;\; i = 1, \cdots, n. \end{aligned} \tag{10}$$

The discriminant function $f : \mathcal{X} \to \mathbb{R}$ is formulated as

$$f(\boldsymbol{x}) = \frac{1}{\lambda}\Big(\sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x})\Big).$$

III. The Regularization Path

In this section we derive nonlinear regularization path algorithm for quadratic loss SVM.

A. Optimal Solution and Regularization Parameter

At the optimal, α_i, $i = 1, \cdots, n$, satisfies the following first-order optimality conditions (KKT conditions):

$$\begin{aligned} \frac{\partial W}{\partial \alpha_i} &= -\frac{1}{\lambda}\sum_{j=1}^{n} Q_{ij}\alpha_j + \rho - \alpha_i \\ &= -y_i f(\boldsymbol{x}_i) + \rho - \alpha_i \begin{cases} \geq 0, & \alpha_i = h, \\ = 0, & \alpha_i \in (0, h), \\ \leq 0, & \alpha_i = 0. \end{cases} \end{aligned} \tag{11}$$

Using these relationships, we define the following index sets:

$$\begin{aligned} \mathcal{L} &= \{i \mid y_i f(\boldsymbol{x}_i) \leq \rho - h, \;\; \alpha_i = h\}, \\ \mathcal{C} &= \{i \mid y_i f(\boldsymbol{x}_i) = \rho - \alpha_i, \;\; \alpha_i \in (0, h)\}, \\ \mathcal{R} &= \{i \mid y_i f(\boldsymbol{x}_i) \geq \rho, \;\; \alpha_i = 0\}. \end{aligned} \tag{12}$$

The regularization path algorithm keeps track of these sets while the regularization parameter λ is perturbed.

In what follows, the subscription by an index set, such as $\boldsymbol{v}_{\mathcal{C}}$ for a vector $\boldsymbol{v} \in \mathbb{R}^n$, indicates a subvector of $\boldsymbol{v}$ whose elements are indexed by $\mathcal{C}$. Similarly, the subscription by two index sets, such as $\boldsymbol{M}_{\mathcal{C},\mathcal{L}}$ for a matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$, denotes a submatrix whose rows are indexed by $\mathcal{C}$ and the columns are indexed by $\mathcal{L}$. Principal submatrix such as $\boldsymbol{Q}_{\mathcal{C},\mathcal{C}}$ is abbreviated as $\boldsymbol{Q}_{\mathcal{C}}$.

The KKT conditions (11) for $i \in \mathcal{C}$ can be written as

$$\sum_{j \in \mathcal{C}} Q_{ij}\alpha_j + \lambda\alpha_j = \rho\lambda - h\sum_{j \in \mathcal{L}} Q_{ij}, \;\; i \in \mathcal{C}.$$

Using matrix notation, it is written as

$$\Big(\boldsymbol{Q}_{\mathcal{C}} + \lambda \boldsymbol{I}\Big)\boldsymbol{\alpha}_{\mathcal{C}} = \rho\lambda \mathbf{1} - h\boldsymbol{Q}_{\mathcal{C},\mathcal{L}}\mathbf{1}, \tag{13}$$

where (i,j)-th entry of $\boldsymbol{Q}$ is Q_{ij} and $\boldsymbol{I}$ is an identity matrix with appropriate size. Let the eigenvalue decomposition (EVD) of $\boldsymbol{Q}_{\mathcal{C}}$ be $\boldsymbol{Q}_{\mathcal{C}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$, where $\boldsymbol{\Sigma} \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ is a diagonal matrix whose i-th diagonal entry σ_i is the i-th eigenvalue of $\boldsymbol{Q}_{\mathcal{C}}$ and $\boldsymbol{U} \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ is an orthogonal matrix whose i-th column is the i-th eigenvector of $\boldsymbol{Q}_{\mathcal{C}}$. It is easy to show that the EVD of $\boldsymbol{Q}_{\mathcal{C}} + \lambda\boldsymbol{I}$ is explicitly obtained as

$$\boldsymbol{Q}_{\mathcal{C}} + \lambda\boldsymbol{I} = \boldsymbol{U}(\boldsymbol{\Sigma} + \lambda\boldsymbol{I})\boldsymbol{U}^\top, \tag{14}$$

Using (13) and (14), we can compute $\boldsymbol{\alpha}_{\mathcal{C}}$ as follows:

$$\boldsymbol{\alpha}_{\mathcal{C}} = \boldsymbol{U}(\boldsymbol{\Sigma} + \lambda\boldsymbol{I})^{-1}\boldsymbol{U}^\top\Big\{\rho\lambda\mathbf{1} - h\boldsymbol{Q}_{\mathcal{C},\mathcal{L}}\mathbf{1}\Big\}. \tag{15}$$

Let us denote the index of the set $\mathcal{C}$ as $\mathcal{C} = \{c_1, \cdots, c_{\ell_C}\}$, where $\ell_C = |\mathcal{C}|$. Then, we can write (15) by element-wise notation:

$$\alpha_{c_i} = \sum_{j=1}^{\ell_C}\sum_{k=1}^{\ell_C}\frac{u_{ik}u_{jk}\{\rho\lambda - hq^{\mathcal{L}}_{c_j}\}}{\sigma_k + \lambda},\ i = 1, \cdots, \ell_C,$$

where u_{ij} is a (i,j)-th entry of $\boldsymbol{U}$ and $q^{\mathcal{L}}_i = \sum_{j\in\mathcal{L}} Q_{ij}$. This equation can be reduced to the following form:

$$\alpha_{c_i} = \rho - \sum_{k=1}^{\ell_C}\frac{\zeta_{ik}}{\sigma_k + \lambda},\ i = 1, \cdots, \ell_C, \tag{16}$$

where

$$\zeta_{ik} \quad = \quad u_{ik}\sum_{j=1}^{\ell_C} u_{jk}(\rho\sigma_k + hq^{\mathcal{L}}_{c_j}).$$

Using (16), $y_i f(\boldsymbol{x}_i)$ can be written as

$$y_i f(\boldsymbol{x}_i) = \frac{1}{\lambda}\left(d_i - \sum_{k=1}^{\ell_C}\frac{\omega_{ik}}{\sigma_k + \lambda}\right), \tag{17}$$

where

$$d_i = \rho\sum_{j\in\mathcal{C}} Q_{ij} + hq^{\mathcal{L}}_i \quad \text{and} \quad \omega_{ik} = \sum_{\ell=1}^{\ell_C} Q_{ic_\ell}\zeta_{\ell k}.$$

The above derivation indicates that, if we have complete information on the index sets $\mathcal{L}$, $\mathcal{C}$, and $\mathcal{R}$, the set of model parameters $\{\alpha_i\}_{i=1}^n$ can be represented as a function of the regularization parameter λ. In particular, for a data point $i \in \mathcal{C}$, the corresponding parameter α_i is formulated by a rational function (16).

B. Event Detection

Equation (16) holds only when the indices in the sets $\mathcal{C}$, $\mathcal{L}$ and $\mathcal{R}$ are not changed. The change of these indices is called an *event*, and a λ is called an *event point* if there is an event at λ. Events in our path-following algorithm are categorized into four types in Table I. Each type of events is relevant to the inequality constraints in the definitions of the sets $\mathcal{C}$, $\mathcal{L}$ and $\mathcal{R}$ in (12). In the case of piecewise linear path, event points are easily detected by solving linear equations.

TABLE I
EVENT CATEGORIZATION

Event	The change of index	The change of inequality
type 1	$i \in \mathcal{C}$ migrates to $\mathcal{R}$	$\alpha_i > 0$ to $\alpha_i = 0$
type 2	$i \in \mathcal{C}$ migrates to $\mathcal{L}$	$\alpha_i < h$ to $\alpha_i = h$
type 3	$i \in \mathcal{R}$ migrates to $\mathcal{C}$	$y_i f(\boldsymbol{x}_i) > \rho$ to $y_i f(\boldsymbol{x}_i) = \rho$
type 4	$i \in \mathcal{L}$ migrates to $\mathcal{C}$	$y_i f(\boldsymbol{x}_i) < \rho - h$ to $y_i f(\boldsymbol{x}_i) = \rho - h$

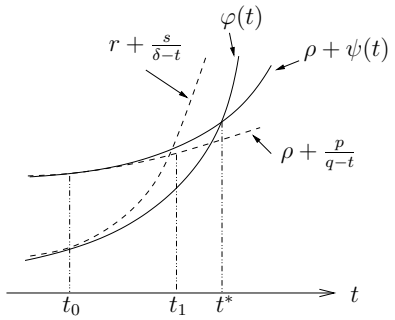


Fig. 3. Rational approximation for the type 1 in Table I. Note that $\rho + \psi_\zeta(t) > \varphi_\zeta(t), t \in (t_0, t^*)$. The approximated solution t_1 can be computed via a quadratic equation. Iterating this, we can obtain a sequence of approximated solution with quadratic convergence.

In our nonlinear path, however, we need to solve nonlinear equations to detect the event points. To this end, we introduce rational approximation approach. Rational approximation has been used in the context of rank-one-update of EVD [8].

Here, we consider how to detect an event point when we decrease λ from the current value λ_0 (the same discussion holds when we increase λ). For the type 1 in Table I, we need to find λ^* such that $\alpha_{c_i} > 0, c_i \in \mathcal{C}$, becomes $\alpha_{c_i} = 0$. Let us define $t = -\lambda$. Then, we increase t from $t_0 = -\lambda_0$ until we find t^* such that

$$\rho - \sum_{k=1}^{\ell_C}\frac{\zeta_{ik}}{\sigma_k - t^*} = 0,\ t^* \in (t_0, 0). \tag{18}$$

If this equation has multiple solutions, we choose the minimum one as t^*. We note the fact that (18) is similar to the *secular equation* which often arises in rank-one-update problem of the EVD [8], [17], [18]. In this paper we introduce *rational approximation* approach [8] for solving (18). Let us define

$$\psi_\zeta(t) \equiv \sum_{k\in\{k|\zeta_{ik}<0\}}\frac{-\zeta_{ik}}{\sigma_k - t},\ \varphi_\zeta(t) \equiv \sum_{k\in\{k|\zeta_{ik}>0\}}\frac{\zeta_{ik}}{\sigma_k - t}.$$

Using these functions, (18) is represented as

$$\rho + \psi_\zeta(t^*) = \varphi_\zeta(t^*),\ t^* \in (t_0, 0).$$

Since the kernel matrix $\boldsymbol{Q}$ is positive semi-definite, $\sigma_k \geq 0$, $k = 1, \cdots, \ell_C$. Therefore, we see that both $\rho + \psi_\zeta(t)$ and $\varphi_\zeta(t)$ are increasing convex functions of $t \in (t_0, 0)$. We approximate ψ_ζ and φ_ζ by their lower and upper bounds (see Fig. 3):

$$\psi_\zeta(t) > \frac{p}{q - t},\ \varphi_\zeta(t) < r + \frac{s}{\delta - t},\ t \in (t_0, 0), \tag{19}$$

where $\delta = \min\{\sigma_k \mid \zeta_{ik} > 0\}$. These upper and lower bound functions are the 1st order local approximations to ψ_ζ and

φ_ζ at t_0, respectively. The four parameters p, q, r and s are defined to satisfy these requirements, *i.e.*, they are defined as

$$\begin{aligned} p &= \frac{\{\psi_\zeta(t_0)\}^2}{\psi'_\zeta(t_0)}, \quad r = \varphi_\zeta(t_0) - \Delta\varphi'_\zeta(t_0), \\ q &= t_0 + \frac{\psi_\zeta(t_0)}{\psi'_\zeta(t_0)}, \quad s = \Delta^2\varphi'_\zeta(t_0), \end{aligned} \tag{20}$$

where $\psi'_\zeta(t) = \partial\psi_\zeta(t)/\partial t$, $\varphi'_\zeta(t) = \partial\varphi_\zeta(t)/\partial t$, and $\Delta = \delta - t_0$. Then, inequalities (19) hold as in [8]. Using the approximation by these simple rational functions in (19), we compute an approximate solution t_1 by solving

$$\rho + \frac{p}{q - t_1} = r + \frac{s}{\delta - t_1}.$$

This equation can be reduced to a quadratic equation, and we choose the minimum one in $(t_0, 0)$ as t_1. In case we have no solution in $(t_0, 0)$, the point $i \in \mathcal{C}$ is disregarded for the moment because it has no chance to define the next event point. Given t_1, we iterate the same procedure to obtain new approximate solution $t_2 \in (t_1, 0)$. As a consequence, we can produce a sequence $\{t_k\}$ that approaches t^* from the left.

For the type 2 in Table I, we need to find λ^* such that $\alpha_{c_i} < h, c_i \in \mathcal{C}$, becomes $\alpha_{c_i} = h$. We increase t from t_0 until we find t^* such that

$$\rho - \sum_{k=1}^{\ell_C} \frac{\zeta_{ik}}{\sigma_k - t^*} = h,\ t^* \in (t_0, 0).$$

This can be written as

$$\rho + \psi_\zeta(t^*) = \phi_\zeta(t^*) + h,\ t^* \in (t_0, 0),$$

We use the following bounds:

$$\psi_\zeta(t) < r + \frac{s}{\delta - t},\ \varphi_\zeta(t) > \frac{p}{q - t},\ t \in (t_0, 0),$$

where $\delta = \min\{\sigma_k \mid \zeta_{ik} < 0\}$. Since $\rho + \psi_\zeta(t) < \varphi_\zeta(t) + h,\ t \in (t_0, t^*)$, we need to set the upper bound to ψ_ζ and the lower bound to φ_ζ. Note that p, q, r and s are computed by alternating ψ_ζ and φ_ζ in (20):

$$\begin{aligned} p &= \frac{\{\varphi_\zeta(t_0)\}^2}{\varphi'_\zeta(t_0)}, \quad r = \psi_\zeta(t_0) - \Delta\psi'_\zeta(t_0), \\ q &= t_0 + \frac{\varphi_\zeta(t_0)}{\varphi'_\zeta(t_0)}, \quad s = \Delta^2\psi'_\zeta(t_0). \end{aligned}$$

For the type 3 of Table I, we have to detect λ^* such that $y_i f(\boldsymbol{x}_i) > \rho$, $i \in \mathcal{R}$, becomes $y_i f(\boldsymbol{x}_i) = \rho$. Using (17), we obtain the following equation:

$$d_i - \sum_{k=1}^{\ell_C} \frac{\omega_{ik}}{\sigma_k - t^*} = -\rho t^*,\ t^* \in (t_0, 0). \tag{21}$$

As in the previous two types of cases, we define the following functions:

$$\psi_\omega(t) \equiv \sum_{k \in \{k \mid \omega_{ik} < 0\}} \frac{-\omega_{ik}}{\sigma_k - t},\ \varphi_\omega(t) \equiv \sum_{k \in \{k \mid \omega_{ik} > 0\}} \frac{\omega_{ik}}{\sigma_k - t}.$$

Then, (21) can be written as

$$d_i + \psi_\omega(t^*) + \rho t^* = \varphi_\omega(t^*),\ t^* \in (t_0, 0). \tag{22}$$

Both sides of equation are increasing convex functions of $t \in (t_0, 0)$. Since $d_i + \psi_\omega(t) + \rho t > \varphi_\omega(t)$, $t \in (t_0, t^*)$, we replace ψ_ω by its lower bound and φ_ω by its upper bound:

$$d_i + \frac{p}{q - t_1} + \rho t_1 = r + \frac{s}{\delta - t_1}, \tag{23}$$

where $\delta = \min\{\sigma_k \mid \omega_{ik} > 0\}$. p, q, r and s are computed by (20) with ψ_ζ replaced by ψ_ω and φ_ζ replaced by φ_ω. We can easily solve (23) because it is reduced to a cubic equation. To detect the event, we only need the minimum solution of that cubic equation in $(t_0, 0)$. Once we obtain t_1, we can iterate rational approximation in the same way as the type 1 and 2.

For the last type 4 of Table I, we solve

$$d_i - \sum_{k=1}^{\ell_C} \frac{\omega_{ik}}{\sigma_k - t} = -(\rho - h)t^*,\ t^* \in (t_0, 0),$$

to find λ^* where $y_i f(\boldsymbol{x}_i) < \rho - h$, $i \in \mathcal{L}$, reaches a boundary $y_i f(\boldsymbol{x}_i) = \rho - h$. This can be written as

$$d_i + \psi_\omega(t^*) + (\rho - h)t^* = \varphi_\omega(t^*),\ t^* \in (t_0, 0). \tag{24}$$

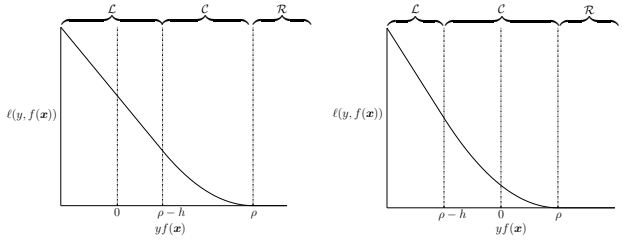
When $h \leq 1$, both sides of the equation are increasing convex functions of $t \in (t_0, 0)$. Even if $h > 1$, we can make the increasing convex functions by moving the term $(\rho - h)t^*$ to the right hand side. In this case, we approximate ψ_ω by its upper bound and φ_ω by its lower bound using the rational approximation.

By checking all the four types in Table I, we obtain λ^*'s for each inequality constraint. The event point is determined as the maximum λ^* among those candidates.

C. Advantages of The Rational Approximation and Cutoff Strategy

Rational approximation generates approximate solution sequence $\{t_k\}$ for a nonlinear equation expressed by two monotonic convex functions. The sequence $\{t_k\}$ can reach $t_k \in [t^* - \varepsilon, t^*]$ by the finite number of iterations for arbitrary small $\varepsilon > 0$. This convergence is guaranteed for any starting point $t_0 < 0$ (note, in contrast, that Newton method may be trapped in periodic cycle). Furthermore, if we assume that the gradient of nonlinear equation is not 0 at t^*, we can prove quadratic convergence of the approximation (we provide the proof in Appendix).

Since $\{t_k\}$ is a monotonically increasing sequence, it approaches t^* from the left. Exploiting this property, we can sometimes terminate iteration for approximation before convergence without affecting the accuracy of the event detection. Suppose we have obtained $\lambda^*_{\max} = -t^*_{\min}$ which is the maximum λ^* among some of the inequalities in Table I. When we investigate the next inequality, we can terminate the approximation at the i-th iteration if t_i becomes larger than $t^*_{\min}$. This is because we only need the minimum t^* to detect the event. We refer to this early termination strategy as *cutoff* (Note that this strategy has no effects on the accuracy of solutions).



(a) case (i): $0 < \rho - h$ (b) case (ii): $0 > \rho - h > -\infty$

Fig. 4. The loss function and $\rho - h$

D. Empty Set $\mathcal{C}$ and Initialization

When we have no data points in $\mathcal{C}$, α_i is either $\alpha_i = 0$, $i \in \mathcal{R}$, or $\alpha_i = 1$, $i \in \mathcal{L}$. Then, $y_i f(\boldsymbol{x}_i)$ can be written as

$$y_i f(\boldsymbol{x}_i) = \frac{h}{\lambda}\Big(\sum_{j \in \mathcal{L}} Q_{ij}\Big).$$

Using this, we can easily check the type 3 and 4 in Table I.

We can use the optimal solution $\boldsymbol{\alpha}$ for any $\lambda > 0$ as a starting point of the regularization path. Although optimal $\boldsymbol{\alpha}$ at initial λ can be obtained by directly solving the optimization problem (10) using, for instance, active set method there is a more appealing approach for initialization. We can find a trivial solution for sufficiently large λ, and we may easily obtain the initial solution by following the path with decreasing λ. We explain how to obtain those trivial solutions in the following three cases: (i) $0 < \rho - h$, (ii) $0 > \rho - h > -\infty$, (iii) $h = \infty$ (squared hinge loss).

When $\lambda = \infty$, optimal $\boldsymbol{w}$ is obviously $\mathbf{0}$, and then $y_i f(\boldsymbol{x}_i) = 0$ for $i = 1, \cdots, n$. Thus, in the first case (i) $0 < \rho - h$, all the data points are in $\mathcal{L}$ (see Fig. 4(a)). We search the first event point λ_1 so that $i \in \mathcal{L}$ moves to $\mathcal{C}$ using the same approach to empty $\mathcal{C}$.

In the second case (ii), all the data points are in $\mathcal{C}$ as $\lambda \to \infty$ (see Fig. 4(b)). Then $y_i f(\boldsymbol{x}_i)$ must be in the following range:

$$y_i f(\boldsymbol{x}_i) \in [\rho - h, \rho], \quad i = 1, \cdots, n. \tag{25}$$

On the other hand, from $\alpha_i \in [0, h]$, we know $y_i f(\boldsymbol{x}_i)$ has the following bounds:

$$y_i f(\boldsymbol{x}_i) = \frac{1}{\lambda}\sum_{j=1}^{n} Q_{ij}\alpha_j \in \frac{h}{\lambda}[\sum_{j=1}^{n} \min(0, Q_{ij}), \sum_{j=1}^{n} \max(0, Q_{ij})]$$

If the inequalities

$$\begin{aligned} \rho - h &\leq \frac{h}{\lambda}\sum_{j=1}^{n} \min(0, Q_{ij}), \quad i = 1, \cdots, n, \\ \rho &\geq \frac{h}{\lambda}\sum_{j=1}^{n} \max(0, Q_{ij}), \quad i = 1, \cdots, n, \end{aligned}$$

are hold, optimal solution of such λ satisfies (25). We can easily calculate λ which satisfies the above inequalities.

In the third case (iii), as in the previous case, all the data points are in $\mathcal{C}$ as $\lambda \to \infty$. However since α_i has no upper bound, we can not apply the same strategy as the previous case. We use the following lower bound:

$$\alpha_{c_i} \;=\; \rho - \sum_{k=1}^{\ell_C} \frac{\zeta_{ik}}{\sigma_k + \lambda} \;\geq\; \rho - \sum_{k \in \{k | \zeta_{ik} > 0\}} \frac{\zeta_{ik}}{\sigma_{\min} + \lambda},$$

where $\sigma_{\min} = \min\{\sigma_k \mid k = 1, \cdots, \ell_C\}$. Since this lower bound monotonically approaches to ρ as $\lambda \to \infty$, all α_{c_i}'s are positive at some large λ. We can find such λ by the following:

$$\max\left\{ \frac{\sum_{k \in \{k | \zeta_{ik} > 0\}} \zeta_{ik}}{\rho} - \sigma_{\min} \,\middle|\, i = 1, \cdots, \ell_C \right\}.$$

If we set $\rho = h$, then $y_i f(\boldsymbol{x}_i) = 0$ is a boundary between $\mathcal{L}$ and $\mathcal{C}$. In this case, detecting the first event is more difficult than the previous three cases. However, even if we can not find trivial initial solution, we can start our path algorithm from any λ and its optimal solution.

E. Computational Complexity

The major computational cost of each iteration of the regularization path involves

- The eigenvalue decomposition of $\ell_C \times \ell_C$ matrix $\boldsymbol{Q}_{\mathcal{C}}$ with $O(\ell_C^3)$.
- Computing $\zeta_{ik}, i = 1, \cdots, \ell_C, k = 1, \cdots, \ell_C$. This needs $O(\ell_C^2)$ computations.
- Computing $\omega_{ik}, i \in \mathcal{R} \cup \mathcal{L}, k = 1, \cdots, \ell_C$. Since we need to compute $\sum_{\ell=1}^{\ell_C} Q_{ic_\ell}\zeta_{\ell k}$ for each ω_{ik}, it takes $O(\ell_C^2(\ell_R + \ell_L))$, where $\ell_R = |\mathcal{R}|$ and $\ell_L = |\mathcal{L}|$.
- Solving nonlinear equations to detect the event. We solve $2\ell_C + \ell_R + \ell_L = O(n)$ equations using the rational approximation. In each iteration of the rational approximation, we have $O(\ell_C)$ computation (mainly for re-calculating α_i or $y_i f(\boldsymbol{x}_i)$). If we assume that the rational approximation terminates at the I_{ra}-th iteration, the total cost is roughly $O(I_{ra}\ell_C n)$.

Thus, the approximate complexity for one iteration of the regularization path is $O(\ell_C^2 n + I_{ra}\ell_C n)$. Note that ℓ_C changes each iteration. If we assume I_{ra} is small enough, it can be considered as $O(\ell_C^2 n)$. As we will see in later experiments, the rational approximation converges very quickly. $O(\ell_C^2 n)$ is similar to the cost of a single SVM training. Therefore, an event detection in our algorithm is as costly as re-training the SVM at each event point. However, empirical results in the next section suggests that our algorithm is an order of magnitude faster than exhaustive grid search by the the SMO algorithm [31], [39]. Note that we take L grid points to run the SMO where L is the number of breakpoints (details are in the next section).

IV. Experiments

To demonstrate our algorithm, we show some numerical results on artificial and real data sets. Using our algorithm, we traced the sequence of event points $\lambda_1 > \lambda_2 > \cdots > \lambda_L$ where λ_1 is the first event point computed by the initialization in Subsection III-D and λ_L is the first event point which becomes smaller than 10^{-5}. The Gaussian RBF kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$ is used. We set kernel

parameter as $\gamma = 1/d$, where d is the number of features (This is a default setting in LIBSVM [10]). In the event detection, we iterate rational approximation until approximation error becomes less than 10^{-12}. We investigated the performances of our algorithm for modified Huber loss function and squared hinge loss. For the former loss function $(\rho, h) = (1.1, 0.2)$ and for the latter loss function $(\rho, h) = (1, \infty)$.

We compared the CPU time of our algorithm with the SMO (Sequential Minimal Optimization) algorithm [31]. Since we did not use the explicit bias term for simplicity, the dual problem (10) has no equality constraints. Then, the SMO algorithm is adapted to optimize only one parameter α_i per iteration [39]. We select updating index i by

$$i = \begin{cases} i_{\rm up}, & \text{if} \quad g_{i_{\rm up}} > -g_{i_{\rm down}}, \\ i_{\rm down}, & \text{if} \quad -g_{i_{\rm down}} > g_{i_{\rm up}}, \end{cases}$$

where $g_i = \partial W / \partial \alpha_i$ and

$$i_{\rm up} = \underset{j \in \{j | \alpha_j < 1\}}{\rm argmax} \; g_j, \; i_{\rm down} = \underset{j \in \{j | \alpha_j > 0\}}{\rm argmax} \; -g_j.$$

The SMO algorithm stops when $|g_i| < 10^{-6}$ are satisfied. We confirmed that the solutions which are obtained by our λ-path algorithm satisfied this condition at all of the breakpoints. We ran the SMO algorithm at L regularization parameters $\lambda^{-1} = \{10^{p_1}, \cdots, 10^{p_L}\}$ where L is the number of the events in regularization path. We set $p_1 = \log_{10} \lambda_1^{-1}$ and uniformly took L values from $[p_1, 5]$. We used the alpha seeding approaches in the SMO, i.e., solution at the previous C is used to produce initial estimates of α_i's. We examined *direct alpha reuse* and *scaling all alphas* strategies (see [13] for detail).

Our regularization path algorithm was mostly implemented in C++. For efficient matrix computations (e.g. matrix vector multiplication or the eigenvalue decomposition), we used LAPACK [2] routine. On the other hand, the SMO algorithm was written solely by C++ on the basis of the-state-of-the-art SVM solver LIBSVM [10]. In both algorithms, we computed and cached the entire kernel matrix at the beginning.

A. Artificial Data

First, we used simple artificial data set. We generated data points $(\boldsymbol{x}, y) \in \mathbb{R}^2 \times \{+1, -1\}$ using the 2-dimensional Normal distributions:

$$\begin{aligned} p(\boldsymbol{x} \mid y = +1) &= \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1^+, \boldsymbol{\Sigma}_1^+) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_2^+, \boldsymbol{\Sigma}_2^+), \\ p(\boldsymbol{x} \mid y = -1) &= \mathcal{N}(\boldsymbol{\mu}^-, \boldsymbol{\Sigma}^-), \end{aligned}$$

where,

$$\begin{aligned} \boldsymbol{\mu}_1^+ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; \boldsymbol{\Sigma}_1^+ = \begin{bmatrix} 0.5 & -0.1 \\ -0.1 & 0.5 \end{bmatrix}, \\ \boldsymbol{\mu}_2^+ &= \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \; \boldsymbol{\Sigma}_2^+ = \begin{bmatrix} 0.5 & 0.1 \\ 0.1 & 0.5 \end{bmatrix}, \\ \boldsymbol{\mu}^- &= \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; \boldsymbol{\Sigma}^- = \begin{bmatrix} 0.5 & -0.1 \\ -0.1 & 0.5 \end{bmatrix}. \end{aligned}$$

We generated $n \in \{100, 200, 400\}$ training data points and the sizes of each class is set to be $n/2$. We normalized each dimension of $\boldsymbol{x}$ to $[0, 1]$. Fig.5 shows an example of data set when $n = 400$. For each size n, we generated 10 data sets to alleviate random sampling effect and computed results as average of 10 runs.

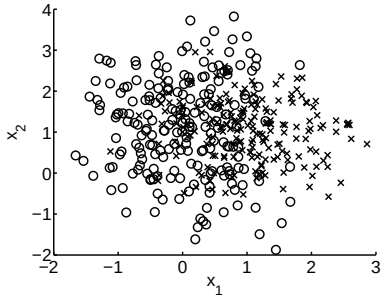


Fig. 5. An example of artificial data set

TABLE III
THE NUMBER OF THE EVENTS AND THE MEAN NUMBER OF ITERATIONS OF A RATIONAL APPROXIMATION OF THE MODIFIED HUBER SVM ($\rho = 1.1, h = 0.2$) FOR ARTIFICIAL DATA. (THE FIGURES IN THE TABLE ARE THE AVERAGE (AND THE STANDARD DEVIATION) OF 10 RUNS)

n	#events L	iteration	iteration (no cutoff)
100	150.00 (14.79)	1.20 (0.04)	11.99 (0.51)
200	286.80 (15.45)	1.12 (0.01)	14.72 (0.46)
400	532.80 (33.02)	1.07 (0.01)	16.28 (0.44)

Table II and III show the results of the modified Huber SVM. Table II compares the CPU times on modified Huber SVM, where figures in the table are the average (and the standard deviations in the round bracket) of 10 runs. We refer to our regularization path algorithm as λ-path in Table II. We see that λ-path is much faster than the SMO algorithm (we observed, as is well known, the SMO took relatively longer time when $C = \lambda^{-1}$ was large [7] (data not shown)). When we did not use cutoff strategy, λ-path becomes much slower. This result suggests that the cutoff strategy can significantly reduce the iterations of rational approximations.

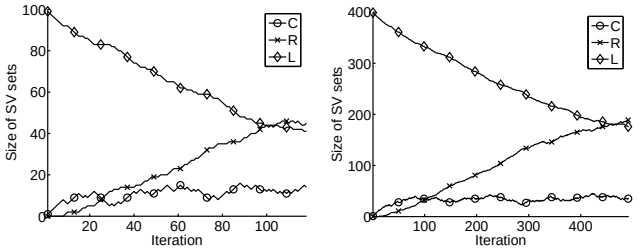
Table III shows the number of the events L and the mean number of iterations in a rational approximation per one nonlinear equation. Some authors suggested that the number of the events appears to be roughly proportional to the number of training points [19], [21], [41]. Although this is only from empirical observations, we also see that the number of the events increased linearly with n in this simple artificial data sets. Even if we did not use cutoff strategy, iterations of rational approximation were only about 10-15 iterations. This rapid convergence is due to the quadratic convergence property of the rational approximation. With the use of cutoff strategy, the average number of iterations became close to 1. This drastic reduction of the number of iterations leads to acceleration of our path algorithm in Table II.

Fig. 6 shows how the sizes of index sets $\mathcal{C}, \mathcal{R}$ and $\mathcal{L}$ change in the λ-path. Each plot is one of the 10 runs of $n = 100$ and 400. The two plots for $n = 100$ and $n = 400$ look similar except their scale.

Table IV and V show the results of the squared hinge SVM. We see similar results to the modified Huber SVM case. Our λ-path algorithm is faster than the SMO. Since the squared

TABLE II
COMPUTATIONAL COST OF THE MODIFIED HUBER SVM ($\rho = 1.1, h = 0.2$) FOR ARTIFICIAL DATA (SEC.)

n	λ-path	λ-path (no cutoff)	SMO (from scratch)	SMO (direct alpha reuse)	SMO (scaling all alphas)
100	0.28 (0.03)	2.80 (0.33)	68.44 (20.41)	49.40 (16.08)	37.73 (12.27)
200	1.10 (0.07)	13.43 (1.08)	548.36 (105.30)	358.15 (80.13)	271.89 (59.33)
400	4.72 (0.33)	57.06 (2.87)	4021.58 (535.73)	2285.46 (261.51)	1635.52 (224.54)



(a) $n = 100$ (b) $n = 400$

Fig. 6. The sizes of index sets $\mathcal{L}, \mathcal{C}$ and $\mathcal{R}$ in the regularization path for artificial data.

TABLE V
THE NUMBER OF THE EVENTS AND THE MEAN NUMBER OF ITERATIONS OF A RATIONAL APPROXIMATION OF THE SQUARED HINGE SVM ($\rho = 1, h = \infty$) FOR ARTIFICIAL DATA. (THE FIGURES IN THE TABLE ARE THE AVERAGE (AND THE STANDARD DEVIATION) OF 10 RUNS)

n	#events L	iteration	iteration (no cutoff)
100	42.10 (8.80)	1.82 (0.21)	14.28 (1.39)
200	83.40 (16.59)	1.40 (0.15)	14.15 (1.74)
400	145.10 (18.58)	1.37 (0.12)	12.92 (1.01)

hinge SVM does not have the set $\mathcal{L}$, the number of event L reduced from the modified Huber SVM case.

B. Real Data

We also apply our algorithm to 6 real world data sets in Table VI. These data sets are available from LIBSVM site [10]. In all data sets, each dimension of $\boldsymbol{x}$ is normalized to $[-1, 1]$. We randomly sampled n data points from original data set 10 times (we set n be approximately 80% of the original number of data points).

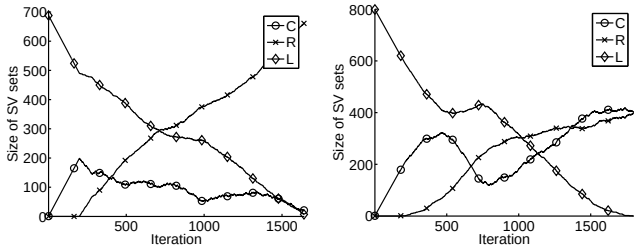
Table VII and VIII show the results of the modified Huber SVM. Table VII shows the CPU time of each algorithm. Our algorithm is much faster than the SMO algorithm in all the data sets. Table VIII shows the number of the events L and the mean number of iterations in a rational approximation. We see that the number of the events is about 2-3 times n and the iteration of rational approximations is very small. These tendencies are also observed in the artificial data set experiments. These

TABLE VI
REAL DATA SETS (THE FIGURES IN THE PARENTHESES ARE THE SIZE OF ORIGINAL DATA SET)

	n	d
sonar	166 (208)	60
heart	216 (270)	13
australian	552 (690)	14
diabetes	614 (768)	8
fourclass	689 (862)	2
german	800 (1000)	24

TABLE VIII
THE NUMBER OF THE EVENTS AND THE MEAN NUMBER OF ITERATIONS OF THE RATIONAL APPROXIMATION OF THE MODIFIED HUBER SVM ($\rho = 1.1, h = 0.2$) FOR REAL DATA SETS.

	#events L	iteration
sonar	279.40 (9.55)	1.14 (0.01)
heart	424.20 (7.05)	1.09 (0.01)
australian	1274.90 (30.90)	1.05 (0.00)
diabetes	1298.80 (27.32)	1.08 (0.00)
fourclass	1634.00 (14.21)	1.04 (0.00)
german	1821.80 (25.07)	1.04 (0.01)



(a) fourclass (b) german

Fig. 7. The size of index sets $\mathcal{L}, \mathcal{C}$ and $\mathcal{R}$ in the regularization path for real data sets.

experimental results illustrate that our approach can trace the nonlinear regularization path more efficiently as well as more precisely than existing rough approximation approaches.

Fig. 7 shows the size of index sets of fourclass and german data sets. Although the size n of these 2 data sets are not much different, the changing patterns of the set sizes are very different. These differences have effect on the computational cost of the regularization path (see Section IV). For example, we need to compute EVD of $\ell_{\mathcal{C}} \times \ell_{\mathcal{C}}$ matrix at each iteration. The german data set has much larger $\ell_{\mathcal{C}}$ compared to the fourclass data set in Fig. 7. Therefore, in the german data set, it takes longer time to compute EVD than the case of the fourclass data set.

Table IX and X are the results of the squared hinge SVM. Here again, we obtain similar results to the modified Huber SVM case. The results demonstrate efficiency of our algorithm.

C. Monitoring the Path of Model-selection Performance Measure

An important advantage of regularization path approach over grid search is that the path of model-selection performance measures (e.g., the path of 0-1 classification loss on validation set) can also be obtained precisely with little additional cost.

Remark 1: In each nonlinear segment between two nearby breakpoints, the model-selection performance measure can be

TABLE IV
COMPUTATIONAL COST OF THE SQUARED HINGE SVM ($\rho = 1, h = \infty$) FOR ARTIFICIAL DATA (SEC.)

n	λ-path	λ-path (no cutoff)	SMO (from scratch)	SMO (direct alpha reuse)	SMO (scaling all alphas)
100	0.18 (0.03)	0.70 (0.22)	185.59 (28.89)	178.05 (28.66)	130.13 (22.48)
200	1.48 (0.20)	4.25 (1.35)	1302.44 (201.31)	1207.30 (186.03)	852.27 (93.04)
400	16.27 (1.37)	28.43 (3.55)	8763.64 (966.75)	7669.12 (824.68)	4962.62 (509.71)

TABLE VII
COMPUTATIONAL COST OF THE MODIFIED HUBER SVM ($\rho = 1.1, h = 0.2$) FOR REAL DATA (SEC.)

	λ-path	SMO (from scratch)	SMO (direct alpha reuse)	SMO (scaling all alphas)
sonar	1.23 (0.08)	9.24 (0.78)	5.25 (0.35)	6.21 (0.57)
heart	2.62 (0.14)	28.97 (6.69)	16.92 (4.55)	20.38 (4.98)
australian	82.81 (7.13)	4294.12 (638.70)	2371.00 (340.46)	2519.12 (366.11)
diabetes	54.01 (3.03)	26987.53 (2189.93)	15310.13 (1320.48)	14316.77 (1274.24)
fourclass	50.22 (1.18)	5019.78 (458.86)	3607.11 (302.86)	2319.08 (210.31)
german	313.91 (15.62)	4118.85 (264.45)	2159.13 (146.79)	2409.60 (164.76)

TABLE IX
COMPUTATIONAL COST OF THE SQUARED HINGE SVM ($\rho = 1, h = \infty$) FOR REAL DATA (SEC.)

	λ-path	SMO (from scratch)	SMO (direct alpha reuse)	SMO (scaling all alphas)
sonar	1.11 (0.03)	3.49 (0.48)	2.69 (0.26)	2.67 (0.29)
heart	2.83 (0.08)	13.75 (2.64)	9.88 (1.78)	10.50 (1.98)
australian	84.61 (2.47)	3498.90 (697.69)	2825.95 (563.69)	2275.22 (395.63)
diabetes	120.69 (4.41)	30884.68 (1550.91)	28118.80 (1344.21)	21276.58 (1041.52)
fourclass	185.34 (6.57)	3403.16 (179.35)	3801.39 (146.05)	1800.37 (132.66)
german	339.70 (9.00)	1383.52 (162.89)	993.90 (138.92)	928.53 (121.83)

TABLE X
THE NUMBER OF THE EVENTS AND THE MEAN NUMBER OF ITERATIONS OF THE RATIONAL APPROXIMATION OF THE SQUARED HINGE SVM ($\rho = 1, h = \infty$) FOR REAL DATA SETS.

	#events L	iteration
sonar	93.30 (4.52)	1.52 (0.09)
heart	146.60 (6.67)	1.41 (0.05)
australian	468.70 (22.32)	1.18 (0.05)
diabetes	353.70 (20.84)	1.22 (0.04)
fourclass	690.00 (12.90)	1.07 (0.01)
german	504.40 (19.52)	1.23 (0.05)

expressed as the function of the regularization parameter λ because the functional form of the solution path $\boldsymbol{\alpha}(\lambda)$ is available.

In this section we illustrate this desirable property by showing the path of 0-1 classification loss and that of AUC (area under the ROC curve). The same experimental setup is employed as section VI-B. For each data set in Table VI, we split the data into two sets: 80% for training and the remaining 20% for validation. The SVM classifier is trained with modified Huber loss $(\rho, h) = (1.1, 0.2)$.

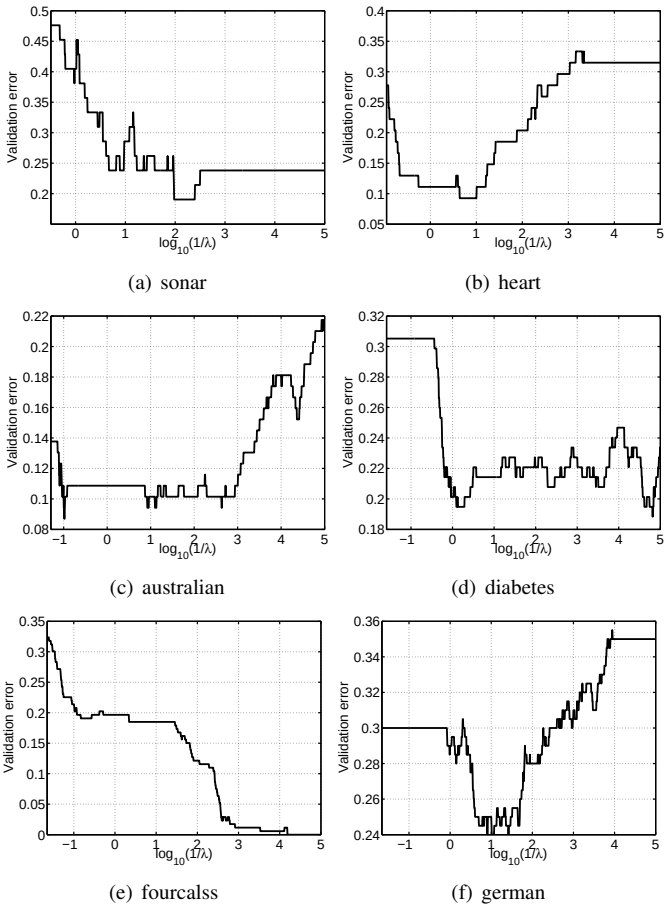
Fig. 9 shows the paths of 0-1 classification loss on the validation sets. We observe in the figure that the paths of the 0-1 loss validation errors have several complicated forms. For example, in (d) diabetes data set, there seems to be two local minimums around $\log_{10}(\frac{1}{\lambda}) \simeq 0.2$ and 4.8. It is difficult to note such detailed observations in grid search approach unless the SVM is trained with huge number of grid of λs.

Fig. 9 shows the paths of AUC on the validation sets. The ROC (Receiver Operating Characteristics) analysis is a standard way to display the rate of true positives against false positives over a range of possible threshold values (e.g., [15]). The AUC (area under the ROC curve) is a natural performance measure for binary classifier. It can be interpreted as the probability that the classifier assigns larger decision function value for a randomly chosen positive example than a random negative example. The plots in the figure indicates that the AUC paths have totally different forms in each data set.

As illustrated in Figs. 8 and 9, the regularization path approach provides the complete picture on how the model-selection performance measure changes with respect to the regularization parameter. On the other hand, the conventional grid search approach only provides the performances in limited number of finite points. The path of model-selection performances as in Figs. 8 and 9 offer additional insight for detailed behavior of the trained classifier (e.g. [14], [38]). Our proposed approach is advantageous especially when such detailed investigations are required.

We also evaluated prediction performance of our path approach. Using 90% of the original data set in Table VI, we performed 10-fold Cross Validation (CV) and remaining 10% was used for the test. As a comparison, we conducted a simple grid search experiments in which the best λ was selected from $\{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ based on CV errors [2]. Table XI and Table XII shows the minimum CV errors and test errors measured in $0-1$ loss function, respectively. Since our path approach monitors changes of the CV error more precisely than the grid search, all of the CV errors of the path algorithm are smaller than the grid search in Table XI. We also see that test performances of our approach in Table XII are comparable or slightly better than the standard grid search.

[2]This the default choice of the well-known LIBSVM software [10].

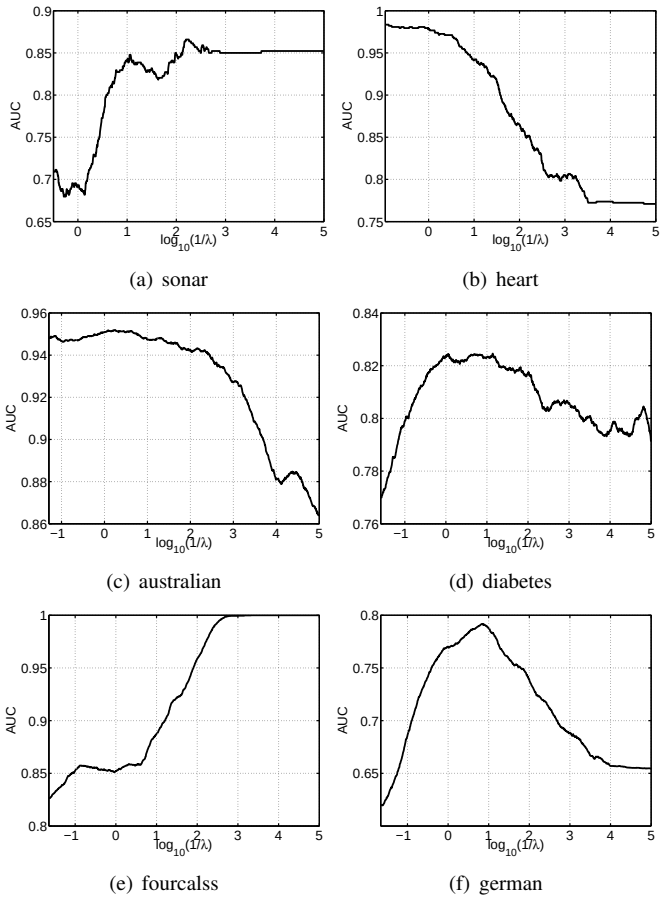


(a) sonar (b) heart

(c) australian (d) diabetes

(e) fourcalss (f) german

Fig. 8. The plots of the validation errors measured by the 0-1 loss.



(a) sonar (b) heart

(c) australian (d) diabetes

(e) fourcalss (f) german

Fig. 9. The plots of the paths of the AUC for validation data set.

TABLE XI
CROSS VALIDATION ERROR COMPARISON $(\rho, h) = (1.1, 0.2)$

	path	grid search
sonar	0.1358	0.1455
heart	0.1584	0.1663
australian	0.1338	0.1386
diabetes	0.2230	0.2271
fourclass	0.0010	0.0022
german	0.2297	0.2348

TABLE XII
TEST ERROR COMPARISON $(\rho, h) = (1.1, 0.2)$

	path	grid search
sonar	0.1227	0.1333
heart	0.1714	0.1630
australian	0.1357	0.1420
diabetes	0.2128	0.2143
fourclass	0.0034	0.0034
german	0.2430	0.2460

V. CONCLUSION

We proposed nonlinear regularization path algorithm for a class of learning machines that have quadratic loss and quadratic penalty which is sometimes referred to as quadratic loss SVM. We developed an accurate and efficient nonlinear path following algorithm using rational approximation technique. Experiments show that the advantage of our algorithm over conventional approach. Since our algorithm uses an EVD at each iteration, we need some further elaborations on the related numerical linear algebra task, especially for applying it to larger data sets.

Another direction of an important future work is to be widening the applicability of regularization path following approach to more recent machine learning techniques such as [20], [27], [42].

REFERENCES

[1] E. L. Allgower and K. Georg, "Continuation and path following," *Acta Numerica*, vol. 2, pp. 1–64, 1993.

[2] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen, *LAPACK Users' guide (third ed.)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999.

[3] F. Bach, D. Heckerman, and E. Horvitz, "Considering cost asymmetry in learning classifiers," *Journal of Machine Learning Research*, vol. 7, pp. 1713–1741, 2006.

[4] F. R. Bach, R. Thibaux, and M. I. Jordan, "Computing regularization paths for learning multiple kernels," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 73–80.

[5] P. L. Bartlett and A. Tewari, "Sparseness vs estimating conditional probabilities: Some asymptotic results," *Journal of Machine Learning Research*, vol. 8, pp. 775–790, 2007.

[6] L. Bo, L. Wang, and L. Jiao, "Recursive finite newton algorithm for support vector regression in the primal," *Neural Comput.*, vol. 19, no. 4, pp. 1082–1096, 2007.

[7] L. Bottou and C.-J. Lin, "Support vector machine solvers," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA.: MIT Press, 2007, pp. 301–320.

[8] J. Bunch, C. Nielsen, and D. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1979.

[9] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., vol. 13. Cambridge, Massachussetts: The MIT Press, 2001, pp. 409–415.
[10] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[11] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.
[12] W. Chu, S. S. Keerthi, and C. J. Ong, "Bayesian support vector regression using a unified loss function," *IEEE Transaction on Neural Networks*, vol. 15, no. 1, pp. 29–44, 2004.
[13] D. DeCoste and K. Wagstaff, "Alpha seeding for support vector machines," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 345–359.
[14] K. Duan, S. S. Keerthi, and A. N. Poo, "Evaluation of simple performance measures for tuning SVM hyperparameters," *Neurocomputing*, vol. 51, pp. 41–59, 2003.
[15] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006.
[16] T. Gal, *Postoptimal Analysis, Parametric Programming, and Related Topics*. Walter de Gruyter, 1995.
[17] G. H. Golub, "Some modified eigenvalue problems," Stanford University, Stanford, CA, USA, Tech. Rep., 1971.
[18] G. H. Golub and C. F. V. Loan, *Matrix computations*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.
[19] L. Gunter and J. Zhu, "Efficient computation and model selection for the support vector regression," *Neural Computation*, vol. 19, no. 6, pp. 1633–1655, 2007.
[20] P. Gutiérrez, C. Hervás-Marti, and F. Martinez-Estudillo, "Logistic regression by means of evolutionary radial basis function neural networks," *Neural Networks, IEEE Transactions on*, vol. 22, no. 2, pp. 246 –263, 2011.
[21] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," *Journal of Machine Learning Research*, vol. 5, pp. 1391–1415, 2004.
[22] M. Karasuyama and I. Takeuchi, "Multiple incremental decremental learning of support vector machines," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds., 2009, pp. 907–915.
[23] S. S. Keerthi and D. DeCoste, "A modified finite newton method for fast solution of large scale linear SVMs," *Journal of Machine Learning Research*, vol. 6, pp. 341–361, 2005.
[24] V. Krishnamurthy, S. D. Ahipasaoglu, and A. d'Aspremont, "A pathwise algorithm for covariance selection," in *NIPS 2009 Workshop on Optimization for Machine Learning*, 2009.
[25] P. Laskov, C. Gehl, S. Kruger, and K.-R. Muller, "Incremental support vector learning: Analysis, implementation and applications," *Journal of Machine Learning Research*, vol. 7, pp. 1909–1936, 2006.
[26] J. Ma and J. Theiler, "Accurate online support vector regression," *Neural Computation*, vol. 15, no. 11, pp. 2683–2703, 2003.
[27] R. Mahdi and E. Rouchka, "Reduced hyperbf networks: Regularization by explicit complexity reduction and scaled rprop-based training," *Neural Networks, IEEE Transactions on*, vol. 22, no. 5, pp. 673–686, 2011.
[28] M. Martin, "On-line support vector machines for function approximation," Software Department, University Politecnica de Catalunya, Tech. Rep., 2002.
[29] C.-J. Ong, S. Shao, and J. Yang, "An improved algorithm for the solution of the regularization path of support vector machine," *Neural Networks, IEEE Transactions on*, vol. 21, no. 3, pp. 451–462, 2010.
[30] M. Park and T. Hastie, "L1-regularization path algorithm for generalized linear models," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 69, no. 4, pp. 659–677, 2007.
[31] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.
[32] K. Ritter, "On parametric linear and quadratic programming problems," in *Mathematical programming: Proceedings of the International Congress on Mathematical Programming*, R. W. Cottle, M. L. Kelmanson, and B. Korte, Eds. Elsevier Science Publishers, 1984, pp. 307–335.
[33] S. Rosset, "Following curved regularized optimization solution paths," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 1153–1160.
[34] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," *Annals of Statistics*, vol. 35, pp. 1012–1030, 2007.
[35] I. Takeuchi, K. Nomura, and T. Kanamori, "Nonparametric conditional density estimation using piecewise-linear solution path of kernel quantile regression," *Neural Computation*, vol. 21, no. 2, pp. 533–559, 2009.
[36] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, no. 1, pp. 267–288, 1996.
[37] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.
[38] V. N. Vapnik and O. Chapelle, "Bounds on error expectation for support vector machines," *Neural Computation*, vol. 12, no. 9, pp. 2013–2036, 2000.
[39] M. Vogt, "SMO algorithms for support vector machines without bias term," Technische Universität Darmstadt, Tech. Rep., 2002.
[40] G. Wang, D.-Y. Yeung, and F. H. Lochovsky, "A kernel path algorithm for support vector machines," in *Twenty-fourth International Conference on Machine Learning*, 2007, pp. 951–958.
[41] ——, "A new solution path algorithm in support vector regression," *Neural Networks, IEEE Transactions on*, vol. 19, no. 10, pp. 1753–1767, 2008.
[42] Y. Washizawa, "Feature extraction using constrained approximation and suppression," *Neural Networks, IEEE Transactions on*, vol. 21, no. 2, pp. 201–210, 2010.
[43] T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization," *Annals of Statistics*, vol. 32, no. 1, pp. 56–134, 2004.

APPENDIX

We provide convergence proof of the rational approximation. We can prove it in almost same way as [8]. Although we will consider the case of solving (22) (i.e., type 3 in Table I), other types can also be proved in similar way. Here, we employ simple notations such as $d = d_i, \psi_k = \psi_\omega(t_k), \psi^* = \psi_\omega(t^*), \psi' = \partial\psi_\omega(t)/\partial t$. In the proof, we assume that the following condition holds:

Assumption 1 $\psi^{*'} + \rho - \varphi^{*'} \neq 0$. Since $\psi^{*'} + \rho - \varphi^{*'} \leq 0$, Assumption 1 just means that $\psi^{*'} + \rho - \varphi^{*'} < 0$.

The following two theorems provide the convergence property of our algorithm.

Theorem 2 *Under Assumption 1, the sequence of the rational approximation $\{t_k\}_{k=1,2,\dots}$ converges to t^* as $k \to \infty$. Even if the Assumption 1 does not hold, $\{t_k\}$ can reach $t^k \in [t^* - \varepsilon, t^*)$ by the finite number of iterations for arbitrary small $\varepsilon > 0$.*

Theorem 3 *Under Assumption 1, if the sequence $\{t_k\}$ converges to t^*, the rational approximation has the quadratic rate of convergence for sufficiently large k.*

A. Proof of Theorem 2

Proof: Let $\beta \in (0, 1)$ be a constant which is independent of the iteration k. We prove the following condition holds for any $t_1 \in (t_0, t^*)$:

$$t^* - t_2 \leq (1 - \beta)(t^* - t_1),$$

where t_2 is obtained by one iteration of the rational approximation from t_1. Let τ satisfy

$$d + \psi_1 + \rho t_1 + (\psi_1' + \rho)(\tau - t_1) = r + \frac{s}{\delta - \tau}.$$

The left hand side represents the tangent line ℓ of $d + \psi(t) + \rho t$ at t_1 (see Fig. 10). Let us define α as the angle between the line ℓ and the horizontal line. Then we see

$$\tan\alpha = \psi_1' + \rho = \frac{r + \frac{s}{\delta - \tau} - (d + \psi_1 + \rho t_1)}{\tau - t_1}.$$

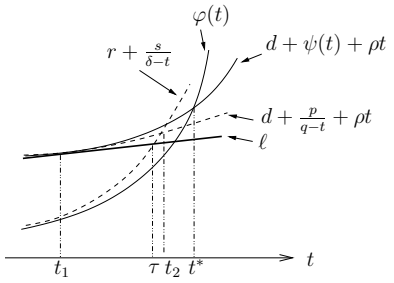


Fig. 10. The schematic illustration of the approximation. ℓ is a tangent line of $d+\psi(t)+\rho t$. From the convexity, ℓ becomes lower bound of $d+\psi(t)+\rho t$ and $d + \frac{p}{q-t} + \rho t$.

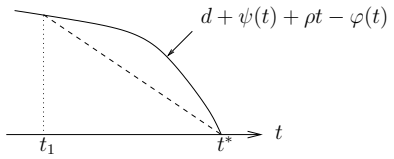


Fig. 11. g_1 is the slope of the dashed line.

From (20), we set $r = \varphi_1 - (\delta - t_1)\varphi_1'$ and $s = (\delta - t_1)^2\varphi_1'$. Substituting r and s into above equation, we obtain

$$\tau - t_1 = -\frac{d + \psi_1 + \rho t_1 - \varphi_1}{\varphi_1' + \rho} + \frac{\varphi_1'}{\psi_1' + \rho}\frac{\delta - t_1}{\delta - \tau}(\tau - t_1).$$

Arranging this equation, we have

$$\tau - t_1 = \frac{-g_1}{\gamma\varphi_1' - \psi_1' - \rho}(t^* - t_1) \tag{26}$$

where $\gamma = (\delta - t_1)/(\delta - \tau)$ and

$$g_1 = -\frac{d + \psi_1 + \rho t_1 - \varphi_1}{t^* - t_1}.$$

g_1 can be interpreted as the slope of a line which passes through $(t_1, d+\psi_1+\rho t_1-\varphi_1)$ and $(t^*, 0)$ (see Fig. 11). From the Assumption 1, g_1 has an upper bound which is smaller than 0, i.e., $g_1 \le g_{\max} < 0$. If the Assumption 1 does not hold, we can maintain this inequality when t_1 is in $(t_0, t^* - \varepsilon]$ for arbitrary small $\varepsilon > 0$. Substituting this into (26), we obtain the following inequalities:

$$\tau - t_1 \ge \frac{-g_{\max}}{\gamma\varphi_1' - \psi_1' - \rho}(t^* - t_1) \ge \beta(t^* - t_1),$$

where

$$\beta = \frac{-g_{\max}}{\max_{t\in(t_0,t^*)}(\gamma\varphi'(t) - \psi'(t) - \rho)}.$$

Since $t_2 \ge \tau$, we have

$$\begin{aligned} t_2 - t_1 &\ge \beta(t^* - t_1) \\ -t^* + t_2 &\ge -(t^* - t_1) + \beta(t^* - t_1) \\ t^* - t_2 &\le (1 - \beta)(t^* - t_1). \end{aligned}$$

Finally, we need to prove $\beta \in (0, 1)$. First, we consider $\beta < 1$. It can be derived from the following inequalities:

$$\begin{aligned} \max_{t\in(t_0,t^*)}(\gamma\varphi'(t) - \psi'(t) - \rho) &> \max_{t\in(t_0,t^*)}(\varphi'(t) - \psi'(t) - \rho) \\ &\ge -g_{\max}. \end{aligned}$$

The first inequality comes from $\gamma > 1$ and $\psi'(t) \ge 0$. From the mean-value theorem, there exists at least one $\theta \in (t_0, t^*)$ such that $\psi'(\theta) + \rho - \varphi'(\theta) = g_{\max}$. Then the second inequality holds. Next, we consider the lower bound of β. From the monotonicity of ψ' and φ', we obtain

$$\max_{t_1\in(t_0,t^*)}(\gamma\varphi_1' - \psi_1' - \rho) < \frac{\delta - t_0}{\delta - t^*}\varphi^{*'} - \psi_0' - \rho.$$

Then we see β is in $(0, 1)$. ■

B. Proof of Theorem 3

Proof: Let κ be a constant which is independent on the iteration. We show $|t_{k+1} - t^*| \le \kappa|t_k - t^*|^2$ for sufficiently large k, when $t_k \to t^*$. Subtracting

$$d + \psi^* + \rho t^* = \varphi^*,$$

from

$$d + \frac{p}{q - t_2} + \rho t_2 = r + \frac{s}{\delta - t_2},$$

we obtain

$$\frac{p}{q - t_2} - \psi^* + \rho(t_2 - t^*) = r + \frac{s}{\delta - t_2} - \varphi^*.$$

Substituting p, q, r and s, this equation can be reduced to

$$\begin{aligned} \psi_1 - \psi^* G(t_1) + \rho(t_2 - t^*)G(t_1) = \\ \left\{\varphi_1 - \varphi^* + \Delta\varphi_1'\left(\frac{t_2 - t_1}{\delta - t_2}\right)\right\}G(t_1), \end{aligned} \tag{27}$$

where $G(t_1) = 1 + \frac{\psi_1'}{\psi_1}(t_1 - t_2)$ and $\Delta = \delta - t_1$. The left hand side of (27) can be written as

$$\begin{aligned} \psi_1 - \psi^* G(t_1) + \rho(t_2 - t^*)G(t_1) = \\ \frac{1}{\psi_1}(\psi_1^2 - \psi^*\psi_1 - \psi^*\psi_1'\varepsilon_1) + \frac{\psi^*\psi_1'}{\psi_1}\varepsilon_2 + \rho\varepsilon_2 G(t_1), \end{aligned}$$

where $\varepsilon_1 = t_1 - t^*$ and $\varepsilon_2 = t_2 - t^*$. Using the Taylor expansion of ψ_1 and ψ_1' around t^*, we obtain

$$\psi_1^2 - \psi^*\psi_1 - \psi^*\psi_1'\varepsilon_1 = \left\{(\psi^{*'})^2 - \frac{1}{2}\psi^*\psi^{*''}\right\}\varepsilon_1^2 + O(\varepsilon_1^3).$$

Finally the left hand side of (27) becomes

$$\frac{\psi^*\psi_1'}{\psi_1}\varepsilon_2 + \rho\varepsilon_2 G(t_1) + O(\varepsilon_1^2). \tag{28}$$

Expanding φ_1 and φ_1' in the right hand side of (27), we obtain

$$\varphi^{*'}\varepsilon_2\left(\frac{\Delta - \varepsilon_1}{\delta - t_2}\right)G(t_1) + O(\varepsilon_1^2). \tag{29}$$

Using (28) and (29), (27) is reduced to

$$\left\{\frac{\psi^*\psi_1'}{\psi_1} + \rho G(t_1) - \varphi_1'\left(\frac{\Delta - \varepsilon_1}{\delta - t_2}\right)G(t_1)\right\}\varepsilon_2 = O(\varepsilon_1^2).$$

Since $G(t_1) \to 1$ as $t_1 \to t^*$,

$$\begin{aligned} \lim_{t_1\to t^*}\left\{\frac{\psi^*\psi_1'}{\psi_1} + \rho G(t_1) - \varphi_1'\left(\frac{\Delta - \varepsilon_1}{\delta - t_2}\right)G(t_1)\right\} = \\ \psi^{*'} + \rho - \varphi^{*'} \ne 0 \end{aligned}$$

Then we can see $\varepsilon_2 = O(\varepsilon_1^2)$. ■