Sequential Minimal Optimization for SVM with Pinball Loss[☆]

Xiaolin Huang[a,*], Lei Shi[b], Johan A.K. Suykens[a]

[a] *KU Leuven, Department of Electrical Engineering (ESAT-STADIUS), B-3001 Leuven, Belgium*
[b] *School of Mathematical Sciences, Fudan University, Shanghai, 200433, P.R. China*

Abstract

To pursue the insensitivity to feature noise and the stability to re-sampling, a new type of support vector machine (SVM) has been established via replacing the hinge loss in the classical SVM by the pinball loss and was hence called a *pin-SVM*. Though a different loss function is used, pin-SVM has a similar structure as the classical SVM. Specifically, the dual problem of pin-SVM is a quadratic programming problem with box constraints, for which the sequential minimal optimization (SMO) technique is applicable. In this paper, we establish SMO algorithms for pin-SVM and its sparse version. The numerical experiments on real-life data sets illustrate both the good performance of pin-SVMs and the effectiveness of the established SMO methods.

Keywords: support vector machine, pinball loss, sequential minimal optimization

1. Introduction

Since proposed in [1] [2], the support vector machine (SVM) has been widely applied and well studied, because of its fundamental statistical property and good generalization capability. The basic idea of SVM is to maximize the margin between two classes by minimizing the regularization term. The margin is classically related to the closest points of two sets, since the hinge loss is minimized. For a given sample set $\mathbf{z} = \{x_i, y_i\}_{i=1}^m$, where $x_i \in \mathbb{R}^n, y_i \in \{-1, +1\}$, the SVM with the hinge loss (C-SVM) in the primal space has the following form,

$$\min_{w,b} \quad \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m} L_{\text{hinge}}\left(1 - y_i(w^T\phi(x_i) + b)\right), \quad (1)$$

where $\phi(x)$ is a feature mapping, $L_{\text{hinge}}(u) = \max\{0, u\}$ is the hinge loss, and C is the trade-off parameter between the margin width and misclassification loss. Since the distance between the closest points is easily affected by the noise on feature x_i, the classifier trained by C-SVM (1) is sensitive to feature noise and unstable to re-sampling. This phenomenon has been observed by many researchers and some techniques have been designed, see, e.g., [3]–[7]. An attractive method for enhancing the stability to feature noise is to change the closest distance measurement to the quantile distance. However, maximizing the quantile distance is non-convex. The well-known ν-support vector machine (ν-SVM, [8]) can be regarded as a convex approach for maximizing the quantile distance and has been successfully applied. In ν-SVM, the margin between the surfaces $\{x : yf(x) = \rho\}$ is maximized. Minimizing the hinge loss together with an additional term $-\nu\rho$ pushes ρ to be the quantile value of $y_i f(x_i)$ and the quantile level is controlled by ν. Recently, we established a new convex method in [9] by extending the hinge loss in C-SVM to the pinball loss. The pinball loss $L_\tau(u)$ is defined as

$$L_\tau(u) = \begin{cases} u, & u \geq 0, \\ -\tau u, & u < 0, \end{cases}$$

which can be regarded as a generalized ℓ_1 loss. Particularly, when $\tau = 0$, the pinball loss $L_\tau(u)$ reduces to the hinge loss. When a positive τ is used, minimizing the pinball loss results in the quantile value. This link has been well studied in quantile regression, see, e.g., [10] [11]. Motivated by this link, the pinball loss with a positive τ value was applied in classification tasks and the related classification method can be formulated as,

$$\min_{w,b} \quad \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m} L_\tau\left(1 - y_i(w^T\phi(x_i) + b)\right), \quad (2)$$

[☆]This work was supported: • EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This paper reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. • Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants • Flemish Government: ○ FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants ○ IWT: projects: SBO POM (100031); PhD/Postdoc grants ○ iMinds Medical Information Technologies SBO 2014 • Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017). L. Shi is also supported by the National Natural Science Foundation of China (11201079) and the Fundamental Research Funds for the Central Universities of China (20520133238, 20520131169). Johan Suykens is a professor at KU Leuven, Belgium.

*Corresponding author.

Email addresses: huangxl06@mails.tsinghua.edu.cn (Xiaolin Huang), leishi@fudan.edu.cn (Lei Shi), johan.suykens@esat.kuleuven.be (Johan A.K. Suykens)

which is called a support vector machine with the pinball loss (*pin-SVM*, [9]). Unlike ν-SVM, pin-SVM pushes the surfaces that define the margin to quantile positions by penalizing also the correctly classified sampling points.

In classification tasks, the pinball loss L_τ has been proved to be calibrated, i.e., the minimizer of the pinball loss has the same sign as $\text{Prob}\{y = +1|x\} - \text{Prob}\{y = -1|x\}$. The preliminary experiments reported in [9] illustrate the stability to feature noise of pin-SVM. A model called *sparse pin-SVM* has been established for enhancing the sparseness. The sparsity is obtained by introducing the ε-zone to the pinball loss, which results in the pinball loss with an ε insensitive zone, denoted by $L_\tau^\varepsilon(u)$:

$$L_\tau^\varepsilon(u) = \begin{cases} u - \varepsilon, & u > \varepsilon, \\ 0, & -\frac{\varepsilon}{\tau} \le u \le \varepsilon, \\ -\tau(u + \frac{\varepsilon}{\tau}), & u < -\frac{\varepsilon}{\tau}. \end{cases} \tag{3}$$

When a training point falls into the interval $[-\frac{\varepsilon}{\tau}, \varepsilon]$, the corresponding dual variable is zero. In Fig.1, we plot $L_\tau^\varepsilon(u)$ for several τ and ε values. When $\varepsilon = 0$, $L_\tau^\varepsilon(u)$ reduces to the pinball loss. Furthermore, if $\tau = 0$, it reduces to the hinge loss.

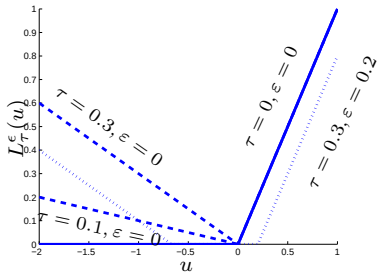


Figure 1: The plots of the pinball loss with an ε insensitive zone. $\tau = 0, \varepsilon = 0$ corresponds to the hinge loss and is displayed by the solid line. When $\varepsilon = 0$, $L_\tau^\varepsilon(u)$ reduces to the pinball loss, as shown by the dashed lines. The dotted line gives the case $\tau = 0.3, \varepsilon = 0.2$.

With properly selected parameters, pin-SVMs can perform better than C-SVM. However, pin-SVMs currently lack fast training algorithms, which is the target of this paper. Generally, we will train pin-SVMs in the dual space by sequential minimal optimization (SMO). SMO is one of the most popular methods for solving SVMs in the dual space. SMO is a kind of decomposition method and always uses the smallest possible working set, which contains two dual variables and can be updated very effectively. For C-SVM, the corresponding SMO algorithms can be found in [12]–[17]. The convergence behavior of SMO has been also well studied in [18]–[22].

In the following, we will first investigate the dual problem of pin-SVM and establish a SMO method in Section 2. Section 3 gives the SMO algorithm for sparse pin-SVM. After that, we use the established SMO algorithms to train pin-SVMs on some real-life problems in Section 4. The numerical experiments confirm the good property of pin-SVM with the proposed methods, which will be promising tools in many applications, as summarized in Section 5.

2. Sequential Minimal Optimization for pin-SVM

2.1. Dual problem of pin-SVM

The dual problem of pin-SVM has been discussed in [9]. In the following, we will first introduce the dual problem and then investigate the problem structure. In the primal space, pin-SVM (2) can be written as the following constrained quadratic programming (QP) problem,

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}w^T w + \sum_{i=1}^m C_i \xi_i \\ \text{s.t.} \quad & y_i \left[w^T \phi(x_i) + b\right] \ge 1 - \xi_i, i = 1, \ldots, m, \\ & y_i \left[w^T \phi(x_i) + b\right] \le 1 + \frac{1}{\tau}\xi_i, i = 1, \ldots, m, \end{aligned} \tag{4}$$

where C_i could be different for different observations. The value of C_i is the weight on the loss related to (x_i, y_i) and one can consider many impacts when setting it. For example, if (x_i, y_i) is an outlier or is heavily noise-polluted, one should choose a small C_i. One noticeable situation is the unbalanced problems, for which the numbers of positive and negative labels are not the same. In this case, we prefer to the following typical setting,

$$\begin{aligned} C_i &= C_0, && \forall i : y_i = 1, \\ C_i &= \frac{\#j:y_j=-1}{\#j:y_j=1} C_0, && \forall i : y_i = -1, \end{aligned} \tag{5}$$

where $C_0 > 0$ is a user-defined constant. In this paper, we always use this setting, which gives equal weights to both classes. The algorithms proposed in the rest of the paper also work for other parameter settings. One can choose suitable C_i according to different applications and prior knowledge.

We introduce the Lagrange multipliers $\alpha_i, \beta_i \ge 0$, which correspond to the constraints in (4). These variables should satisfy the following complementary slackness condition,

$$\begin{aligned} \alpha_i \left(1 - \xi_i - y_i \left[w^T \phi(x_i) + b\right]\right) = 0, \\ i = 1, 2, \ldots, m, \\ \beta_i \left(y_i \left[w^T \phi(x_i) + b\right] - \frac{1}{\tau}\xi_i - 1\right) = 0, \\ i = 1, 2, \ldots, m. \end{aligned}$$

Considering the Lagrangian of (4) and KKT condition, we get the following dual problem for pin-SVM,

$$\begin{aligned} \min_{\alpha,\beta} \quad & \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m (\alpha_i - \beta_i) y_i \mathcal{K}_{ij} y_j (\alpha_j - \beta_j) \\ & - \sum_{i=1}^m (\alpha_i - \beta_i) \\ \text{s.t.} \quad & \sum_{i=1}^m y_i (\alpha_i - \beta_i) = 0 \\ & \alpha_i + \frac{1}{\tau}\beta_i = C_i, i = 1, 2, \ldots, m, \\ & \alpha_i \ge 0, \beta_i \ge 0, i = 1, 2, \ldots, m, \end{aligned} \tag{6}$$

where $\mathcal{K}$ corresponds to a positive definite kernel with $\mathcal{K}_{ij} = \mathcal{K}(x_i, x_j) = \phi(x_i)^T\phi(x_j)$. After obtaining the solution of (6), we use the sign of the following function to do classification:

$$f(x) = \sum_{i=1}^{m} y_i(\alpha_i - \beta_i)\mathcal{K}(x, x_i) + b,$$

where b is computed according to the complementary slackness conditions

$$y_i f(x_i) = 1, \forall i \in \{j : \alpha_j \neq 0, \beta_j \neq 0\}.$$

We further introduce $\lambda_i = \alpha_i - \beta_i$ and eliminate the equality constraint $\alpha_i + \frac{1}{\tau}\beta_i = C_i$. Then the equivalent formulation of (6) can be posed as

$$\begin{aligned} \min_{\lambda} \quad & F(\lambda) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\lambda_i y_i \mathcal{K}_{ij} y_j \lambda_j - \sum_{i=1}^{m}\lambda_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} y_i\lambda_i = 0, \\ & -\tau C_i \leq \lambda_i \leq C_i, i = 1, 2, \ldots, m. \end{aligned} \tag{7}$$

We again observe the relationship between pin-SVM and C-SVM in the dual space: pin-SVM with $\tau = 0$ reduces to C-SVM. The optimization problem (7) is a quadratic programming with box constraints. Therefore, we can update a part of the dual variables and keep the others unchanged, i.e., the sequential minimization optimization (SMO, [12]–[17]) is applicable to train pin-SVM (7).

The constraint $-\tau C_i \leq \lambda_i \leq C_i$ can be equivalently transformed into

$$A_i \leq y_i\lambda_i \leq B_i,$$

where

$$A_i = \begin{cases} -\tau C_i, & y_i = 1, \\ -C_i, & y_i = -1, \end{cases} \qquad B_i = \begin{cases} C_i, & y_i = 1, \\ \tau C, & y_i = -1. \end{cases}$$

For a given λ, the indices are divided into the following two sets,

$$I_{\text{up}}^{\lambda} = \{i : y_i\lambda_i < B_i\} \text{ and } I_{\text{down}}^{\lambda} = \{i : y_i\lambda_i > A_i\}.$$

The subscripts of the two sets imply that for a pair of observations $i \in I_{\text{up}}^{\lambda}, j \in I_{\text{down}}^{\lambda}$, one can always find a small positive scalar t such that the modified solution $\lambda_i + t, \lambda_j - t$ remains feasible. Therefore, if λ is an optimizer, the following inequality should be met

$$y_i g_i^{\lambda} \geq y_j g_j^{\lambda},$$

where

$$g_i^{\lambda} = y_i \sum_{j=1}^{m} y_j\lambda_j\mathcal{K}_{ij} - 1$$

stands for the derivatives of the objective function of (7) with respect to α_i. Otherwise, if $y_i g_i^{\lambda} < y_j g_j^{\lambda}$, we can update λ_i and λ_j to obtain a strict decrease on the objective value of (7). Since the above inequality holds for any $i \in I_{\text{up}}(\lambda)$ and $j \in I_{\text{down}}(\lambda)$, a necessary condition of λ^* being optimal (7) can be written as:

$$\sum_{i=1}^{m} y_i\lambda_i^* = 0,$$

and

$$\exists \rho \in \mathbb{R} \text{ such that } \max_{i \in I_{\text{up}}^{\lambda^*}} y_i g_i^{\lambda^*} \leq \rho \leq \min_{j \in I_{\text{down}}^{\lambda^*}} y_j g_j^{\lambda^*}. \tag{8}$$

The corresponding condition for C-SVM has been widely applied in the SMO technique, see, e.g., [20] and [14]. When τ varies, $I_{\text{up}}^{\lambda^*}$ and $I_{\text{down}}^{\lambda^*}$ are different.

2.2. Dual variable update

Sequential minimal optimization starts from an initial feasible solution of (7) and updates λ until (8) is satisfied. The basic idea of SMO is that we only update the dual variables in a working set and leave the other variables unchanged. The extreme case is that only two variables are involved in each iteration, which follows that there exists an explicit update formulation.

Denote the current solution by λ^{old}. Without any loss of generalization, we assume that $i \in I_{\text{up}}^{\lambda^{\text{old}}}, j \in I_{\text{down}}^{\lambda^{\text{old}}}$ are the variables in the working set. That means that the two elements violate the optimality condition (8), i.e.,

$$y_i g_i^{\lambda^{\text{old}}} > y_j g_j^{\lambda^{\text{old}}}. \tag{9}$$

Denote u^{ij} for a vector of which the i-th component is y_i, the j-th component is $-y_j$, and the others are zero. Then searching along u^{ij} will bring the improvement for (7). Specifically, $\lambda^{\text{old}} + \zeta u^{ij}$ with a sufficiently small positive $\zeta > 0$ will be still feasible to (7). Moreover,

$$\begin{aligned} &F(\lambda^{\text{old}} + \zeta u^{ij}) - F(\lambda^{\text{old}}) \\ &\quad = \zeta\left(y_i g_i^{\lambda^{\text{old}}} - y_j g_j^{\lambda^{\text{old}}}\right) - \tfrac{\zeta^2}{2}(\mathcal{K}_{ii} + \mathcal{K}_{jj} - 2\mathcal{K}_{ij}). \end{aligned} \tag{10}$$

From this formulation and (9), we know that the objective function of (7) can be decreased strictly. The best ζ which gives the largest decrease of the objective function is the minimizer of the following problem,

$$\begin{aligned} \min_{\zeta \geq 0} \quad & -\zeta\left(y_i g_i^{\lambda^{\text{old}}} - y_j g_j^{\lambda^{\text{old}}}\right) + \\ & \frac{\zeta^2}{2}\left(\mathcal{K}_{ii} + \mathcal{K}_{jj} - 2\mathcal{K}_{ij}\right) \\ \text{s.t.} \quad & y_i g_i^{\lambda^{\text{old}}} + \zeta \leq B_i, \\ & y_j g_j^{\lambda^{\text{old}}} - \zeta \geq A_j. \end{aligned}$$

For this 1-dimensional QP, the optimal solution can be explicitly given by

$$\widehat{\zeta} = \min\left\{B_i - y_i g_i^{\lambda^{\text{old}}}, y_j g_j^{\lambda^{\text{old}}} - A_j, \frac{y_i g_i^{\lambda^{\text{old}}} - y_j g_j^{\lambda^{\text{old}}}}{\mathcal{K}_{ii} + \mathcal{K}_{jj} - 2\mathcal{K}_{ij}}\right\}.$$

Correspondingly, the dual variables are updated to

$$\lambda_i^{\text{new}} = \lambda_i^{\text{old}} + \widehat{\zeta} y_i \quad \text{and} \quad \lambda_j^{\text{new}} = \lambda_j^{\text{old}} - \widehat{\zeta} y_j.$$

At the same time, the gradient vector is updated to

$$g_l^{\lambda^{\text{new}}} = g_l^{\lambda^{\text{old}}} - \widehat{\zeta} y_l \mathcal{K}_{il} + \widehat{\zeta} y_l \mathcal{K}_{jl}, \quad \forall l = 1, 2, \ldots, m.$$

2.3. Working set selection and initial solution

Above we discussed the update process for pin-SVM when $i \in I_{\text{up}}^{\lambda^{\text{old}}}, j \in I_{\text{down}}^{\lambda^{\text{old}}}$ are chosen in the working set. Before establishing the SMO for pin-SVM, we first consider the working set selection and initial solution generation.

The objective function of pin-SVM (7) is the same as that of C-SVM. Thus, the strategies of selecting two dual variables for C-SVM are applicable to pin-SVM. The simplest selection is the maximal violating pair, which has been discussed in [20]. For the current solution λ^{old}, we choose i and j as

$$i = \arg \max_{l \in I_{\text{up}}^{\lambda^{\text{old}}}} y_l g_l^{\lambda^{\text{old}}} \quad \text{and} \quad j = \arg \min_{l \in I_{\text{down}}^{\lambda^{\text{old}}}} y_l g_l^{\lambda^{\text{old}}}. \quad (11)$$

This strategy is essentially the greedy choice based on the first order approximation of $F(\lambda^{\text{old}} + \zeta u^{ij}) - F(\lambda^{\text{old}})$. One can also consider the second order working set selection proposed by [13]. That method is based on the second order expansion (10). This quadratic gain should be maximized with the linear constraints. To quickly and heuristically find a good direction, we ignore the constraint and then can find the maximal gain easily:

$$\frac{(y_i g_i^{\lambda^{\text{old}}} - y_j g_j^{\lambda^{\text{old}}})^2}{2(\mathcal{K}_{ii} + \mathcal{K}_{jj} - 2\mathcal{K}_{ij})}. \quad (12)$$

One can choose i, j by maximizing (12) but it needs pairwise comparison. Instead, we first use (11) to find i and then only choose j according to (12), which simply requires element comparison. This is also the strategy utilized for C-SVM in LIBSVM [17].

For the initialization, we use $\lambda_i = -\tau C_i$. Recalling (5) for the setting of C_i, one can verify that $\lambda_i = -\tau C_i$ gives a feasible solution of (7). When $\tau = 0$, the initial solution is $\lambda = 0$, which is commonly used for C-SVM. If we know the optimal solution for pin-SVM with τ_1, denoted by $\lambda^{(\tau_1)}$, then we can have a good guess for pin-SVM with τ_2. To observe the link between $\lambda^{(\tau_1)}$ and $\lambda^{(\tau_2)}$, we illustrate a simple classification task "two moons" in Fig.2, where the red crosses and the green stars correspond to observations in class $+1$ and class -1, respectively. We use pin-SVM (7) to train the classifier. In this example, the same radial basis function (RBF) kernel and the same regularization parameter, but different τ values are used. The surfaces $\{x : f(x) = -1, +1\}$ are displayed in Fig.2.

According to the complementary slackness conditions, we know that

$$\begin{aligned} y_i f(x_i) > 1, &\quad \Rightarrow \quad i \in \mathcal{S}_- = \{j : \lambda_j = -\tau C_j\}, \\ y_i f(x_i) = 0, &\quad \Leftarrow \quad i \in \mathcal{S}_0 = \{j : -\tau C_j < \lambda_j < C_j\}, \\ y_i f(x_i) < 1, &\quad \Rightarrow \quad i \in \mathcal{S}_+ = \{j : \lambda_j = C_j\}. \end{aligned}$$

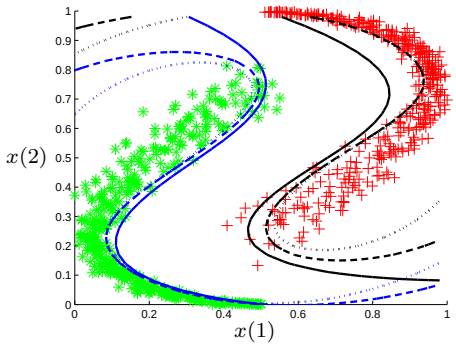


Figure 2: Sampling points and classification results of pin-SVM. Points in class $+1$ and -1 are shown by green stars and red crosses, respectively. The surfaces $\{x : f(x) = 1\}$ (blue lines) and $\{x : f(x) = -1\}$ (black lines) for $\tau = 0, 0.05, 0.1$ are displayed by solid, dash-dotted, and dotted lines, respectively.

In other words, the surfaces $\{x : f(x) = \pm 1\}$ partition the training sets into three parts. Most of the dual variables take the values $-\tau C_i$ or C_i. The left data are located in $\{x : f(x) = +1\}$ or $\{x : f(x) = -1\}$. From Fig.2, we observe that for many points, they are located in the same part for different τ. Fig.2 also illustrates that with the increasing τ, the surfaces $f(x_i) = \pm 1$ move towards the decision boundary. This can be observed as well from the primal form (2), of which the optimality condition can be written as the existence of $\eta_i \in [-\tau, 1]$ such that

$$\frac{w_j}{C} - \sum_{i \in \mathcal{S}_-} y_i \phi_j(x_i) + \tau \sum_{i \in \mathcal{S}_+} y_i \phi_j(x_i) - \sum_{i \in \mathcal{S}_0} \eta_i y_i \phi_j(x_i) = 0, \forall j.$$

This condition implies that generally a larger τ results in more data falling into $\mathcal{S}_-$. Therefore, if $\tau_1 > \tau_2$ and the difference is not big, it is with a high probability that $\lambda_i^{(\tau_2)} = -\tau_2 C_i$ if $\lambda_i^{(\tau_1)} = -\tau_1 C_i$. Following from this discussion, we suggest Algorithm 1 for the initial solution.

By the proposed procedure, we find a new feasible solution, which is heuristically suitable for τ_2. When tuning the parameter τ, we need to train pin-SVM for a series of τ values, for which the above procedure can be applied.

Now we give the SMO algorithm for pin-SVM (1) in Algorithm 2, where e is a pre-defined accuracy and is set to be 10^{-6} in this paper.

3. SMO for Sparse pin-SVM

Pin-SVM can be regarded as an extension to C-SVM via introducing flexibility on τ. Since quantile distances are considered, pin-SVM is insensitive to feature noise and has shown better classification accuracy over C-SVM. In pin-SVM (7), the dual variables are categorized into three types: *lower bounded support vectors* ($\lambda_i = -\tau C_i$), *free support vectors* ($-\tau C_i < \lambda_i < C_i$), and *upper bounded support vectors* ($\lambda_i = C_i$). When $\tau = 0$, pin-SVM reduces to C-SVM. Correspondingly, the lower bounded support vectors are zero and C-SVM has sparseness. To pursue

Algorithm 1: Initialization for pin-SVM with τ_2 from $\lambda^{(\tau_1)}$

Set $\mathcal{S}_-^{\lambda^{(\tau_1)}} := \left\{i : \lambda_i^{(\tau_1)} = -\tau_1 C_i\right\}$,
$\mathcal{S}_+^{\lambda^{(\tau_1)}} := \left\{i : \lambda_i^{(\tau_1)} = C_i\right\}$;
Let $\tilde{\lambda}_i := -\tau_2 C_i, \forall i \in \mathcal{S}_-^{\lambda^{(\tau_1)}}$ and
$\tilde{\lambda}_i := C_i, \forall i \in \mathcal{S}_+^{\lambda^{(\tau_1)}}$;
Calculate the violation $v := \sum_{i=1}^m y_i \tilde{\lambda}_i$;
if $\tau_2 > \tau_1$ **then**
  repeat
    select i from $\{i : y_i = \text{sign}(v)\} \bigcap \mathcal{S}_+^{\lambda^{(\tau_1)}}$;
    set $\tilde{\lambda}_i := \max\{C_i - v, -\tau_2 C_i\}$;
    update $v := \max\{0, v - (1+\tau_2)C_i\}$;
  until $v = 0$;
else
  repeat
    select i from $\{i : y_i = -\text{sign}(v)\} \bigcap \mathcal{S}_-^{\lambda^{(\tau_1)}}$;
    set $\tilde{\lambda}_i := \max\{-\tau_2 C_i + v, C_i\}$;
    update $v := \max\{0, v - (1+\tau_2)C_i\}$;
  until $v = 0$;
end
Return $\tilde{\lambda}$ as the initial solution for pin-SVM with τ_2.

Algorithm 2: SMO for pin-SVM

Set $\lambda_i := -\tau C_i$ or use Algorithm 1 to generate λ ;
Calculate $g_i := y_i \sum_{j=1}^m y_j \lambda_j \mathcal{K}_{ij} - 1$ and set

$$A_i := \begin{cases} -\tau C_i, & y_i = 1 \\ -C_i, & y_i = -1 \end{cases} \quad B_i := \begin{cases} C_i, & y_i = 1 \\ \tau C_i, & y_i = -1 \end{cases};$$

repeat
  $I_{\text{up}}^\lambda := \{i : y_i\lambda_i < B_i\}, I_{\text{down}}^\lambda := \{i : y_i\lambda_i > A_i\}$;
  select $i := \arg\max_{l \in I_{\text{up}}^\lambda} y_l g_l$;
  select $j := \arg\max_{l \in I_{\text{down}}^\lambda} \frac{(y_i g_i^\lambda - y_l g_l^\lambda)^2}{2(\mathcal{K}_{ii} + \mathcal{K}_{ll} - 2\mathcal{K}_{il})}$;
  calculate the update length
    $\zeta := \min\left\{B_i - y_i g_i, y_j g_j - A_j, \frac{y_i g_i - y_j g_j}{\mathcal{K}_{ii} + \mathcal{K}_{jj} - 2\mathcal{K}_{ij}}\right\}$;
  update $\lambda_i := \lambda_i + y_i\zeta$ and $\lambda_j := \lambda_j + y_j\zeta$,
  $g_l := g_l - \zeta y_l \mathcal{K}_{il} + \zeta y_l \mathcal{K}_{jl}, \forall l = 1, \ldots, m$;
until $\max_{i \in I_{\text{up}}^\lambda} y_i g_i - \min_{j \in I_{\text{down}}^\lambda} y_j g_j < e$;
Calculate $b := \frac{1}{2}\left(\max_{i \in I_{\text{up}}^\lambda} y_i g_i + \min_{j \in I_{\text{down}}^\lambda} y_j g_j\right)$.

sparseness for pin-SVM with a nonzero τ value, a loss function with an ε insensitive zone was applied. Then a *sparse pin-SVM* has been established in [9]. In the primal space, sparse pin-SVM can be posed as

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^m L_\tau^\varepsilon\left(1 - y_i(w^T\phi(x_i) + b)\right), \qquad (13)$$

where the pinball loss with an ε insensitive zone $L_\tau^\varepsilon(u)$ is defined in (3). The dual problem of (13) has been deduced in [9] and takes the following form,

$$\begin{aligned} \min_{\lambda,\gamma} \quad & \frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \lambda_i y_i \mathcal{K}_{ij} y_j \lambda_j - \sum_{i=1}^m \lambda_i - \varepsilon\sum_{i=1}^m \gamma_i \\ \text{s.t.} \quad & \sum_{i=1}^m y_i\lambda_i = 0, \qquad (14) \\ & \gamma_i \geq 0, \ i = 1, \ldots, m, \\ & -\tau(C_i - \gamma_i) \leq \lambda_i \leq C_i - \gamma_i, \ i = 1, \ldots, m. \end{aligned}$$

The possible range of the dual variable γ_i is $0 \leq \gamma_i \leq C_i$. When γ_i takes value C_i, the corresponding λ_i will be zero, which brings sparsity to pin-SVM. From the objective function of (14), one can see that a large ε will push γ_i close to C_i, i.e., there are more λ_i values zero.

The last constraint in (14) can be viewed as a box constraint on λ_i and the box depends on another dual variable γ_i. Similarly to the discussion on pin-SVM (7), we can write $-\tau(C_i - \gamma_i) \leq \lambda_i \leq C_i - \gamma_i$ as

$$A_i^{\gamma_i} \leq y_i\lambda_i \leq B_i^{\gamma_i},$$

where

$$A_i^{\gamma_i} = \begin{cases} -\tau(C_i - \gamma_i), & y_i = 1, \\ -(C_i - \gamma_i), & y_i = -1, \end{cases}$$

and

$$B_i^{\gamma_i} = \begin{cases} C_i - \gamma_i, & y_i = 1, \\ \tau(C_i - \gamma_i), & y_i = -1. \end{cases}$$

Then for given γ and λ, we can find the two following sets,

$$I_{\text{up}}^{\lambda,\gamma} = \{i : y_i\lambda_i < B_i^{\gamma_i} \text{ or } \gamma_i > 0\},$$

and

$$I_{\text{down}}^{\lambda,\gamma} = \{i : y_i\lambda_i > A_i^{\gamma_i} \text{ or } \gamma_i > 0\}.$$

Here $\gamma_i > 0$ can guarantee that $\lambda_i \pm \zeta$ is feasible for sufficiently small scalar ζ. Then, necessary conditions for λ, γ being optimal to (14) can be presented as follows:

- for a given γ value, λ should satisfy:

$$\max_{i \in I_{\text{up}}^{\lambda,\gamma}} y_i g_i^\lambda \leq \min_{j \in I_{\text{down}}^{\lambda,\gamma}} y_j g_j^\lambda, \text{ and } \sum_{i=1}^m y_i\lambda_i = 0;$$

- for a given λ value, γ should satisfy:

$$\gamma_i = \min\left\{C_i + \frac{1}{\tau}\lambda_i, C_i - \lambda_i\right\}.$$

Notice that in sparse pin-SVM (14), the gradient g_i^λ is different from that in pin-SVM (6), since there is one additional freedom on γ_i. Specifically, there are three situations. If $\lambda_i = C_i - \gamma_i$, then

$$g_i^\lambda = y_i \sum_{j=1}^{m} y_j \lambda_j \mathcal{K}_{ij} - 1 + \varepsilon.$$

If $\lambda_i = -\tau(C_i - \gamma_i)$, then we have

$$g_i^\lambda = y_i \sum_{j=1}^{m} y_j \lambda_j \mathcal{K}_{ij} - 1 - \frac{\varepsilon}{\tau}.$$

Otherwise, i.e., $-\tau(C_i - \gamma_i) < \lambda_i < C_i - \gamma_i$, we have

$$g_i^\lambda = y_i \sum_{j=1}^{m} y_j \lambda_j \mathcal{K}_{ij} - 1.$$

The above conditions are given separately for λ and γ. For sparse pin-SVM (14), λ_i and γ_i are coupled in the constraints. Hence these conditions are necessary but not sufficient. However, to pursue an efficient solving method for (14), we apply the above necessary condition to choose two data points in a working set. Then the selected dual variables are modified and the others are unchanged.

Similarly to pin-SVM, the working set for sparse pin-SVM (14) contains at least two data points. Suppose that i, j are selected. Then to update $\lambda_{i,j}, \gamma_{i,j}$, we are to solve the following QP problem

$$\begin{aligned}
\min_{\lambda_{i,j},\gamma_{i,j}} \quad & \frac{1}{2}\mathcal{K}_{ii}\lambda_i^2 + \lambda_j y_j \mathcal{K}_{ij} y_j \lambda_j + \frac{1}{2}\mathcal{K}_{jj}\lambda_j^2 \\
& + \lambda_i y_i \sum_{l \neq i,j} y_l \lambda_l \mathcal{K}_{il} + \lambda_j y_j \sum_{l \neq i,j} y_l \lambda_l \mathcal{K}_{jl} \\
& - \lambda_i - \lambda_j - \varepsilon\gamma_i - \varepsilon\gamma_j \\
\text{s.t.} \quad & y_i \lambda_i + y_j \lambda_j = -\sum_{l \neq i,j} y_l \lambda_l, \qquad (15) \\
& \gamma_i \geq 0, \gamma_j \geq 0, \\
& -\tau(C_i - \gamma_i) \leq \lambda_i \leq C_i - \gamma_i, \\
& -\tau(C_j - \gamma_j) \leq \lambda_j \leq C_j - \gamma_j.
\end{aligned}$$

When $\gamma_{i,j}$ are fixed, (15) reduces to a 2-dimensional QP with one equality constraint, which has an explicit solution. This is the case for pin-SVM (7). However, in sparse pin-SVM, $\gamma_{i,j}$ and $\lambda_{i,j}$ are coupled and there is no explicit solution. Hence, we have to solve (15) to update $\lambda_{i,j}, \gamma_{i,j}$ at each iteration.

Solving (15) decreases the objective of (14). We should choose the reasonable working set according to the gain of solving (15). The gain is better than the case keeping $\gamma_{i,j}$ unchanged. For the case $\gamma_{i,j}$ fixed, the gain is (10), from which we can estimate the gain for (15) and then select the working set by the following rule:

$$\begin{aligned}
i &= \arg\max\nolimits_{l \in I_{\text{up}}^{\lambda,\gamma}} y_l g_l^\lambda, \\
j &= \arg\max\nolimits_{l \in I_{\text{down}}^{\lambda,\gamma}} \frac{(y_i g_i^\lambda - y_l g_l^\lambda)^2}{2(\mathcal{K}_{ii} + \mathcal{K}_{ll} - 2\mathcal{K}_{il})}.
\end{aligned}$$

This selection strategy is similar to that for pin-SVM, but now it is dependent on γ. The initial solution for pin-SVM $\lambda_i = -\tau C_i$ is also feasible to sparse pin-SVM (14). Correspondingly, the initial γ is set to be $\gamma_i = \min\left\{C_i + \frac{1}{\tau}\lambda_i, C_i - \lambda_i\right\}$, which is according to the necessary optimal condition.

Now the sequential minimal optimization for sparse pin-SVM (14) is summarized in Algorithm 3.

Algorithm 3: SMO for sparse pin-SVM

Set $\lambda_i := -\tau C_i$ and $\gamma_i := \min\left\{C_i + \frac{1}{\tau}\lambda_i, C_i - \lambda_i\right\}$;
Calculate $g_i := y_i \sum_{j=1}^{m} y_j \lambda_j \mathcal{K}_{ij} - 1$;

$$A_i^{\gamma_i} := \begin{cases} -\tau(C_i - \gamma_i), & y_i = 1 \\ -(C_i - \gamma_i), & y_i = -1 \end{cases};$$

$$B_i^{\gamma_i} := \begin{cases} C_i - \gamma_i, & y_i = 1 \\ \tau(C_i - \gamma_i), & y_i = -1 \end{cases};$$

repeat
  $I_{\text{up}}^{\lambda,\gamma} := \{i : y_i\lambda_i < B_i^{\gamma_i} \text{ or } \gamma_i > 0\}$;
  $I_{\text{down}}^{\lambda,\gamma} := \{i : y_i\lambda_i > A_i^{\gamma_i} \text{ or } \gamma_i > 0\}$;
  select $i := \arg\max_{l \in I_{\text{up}}^{\lambda}} y_l g_l$;
  select $j := \arg\max_{l \in I_{\text{down}}^{\lambda}} \frac{(y_i g_i^\lambda - y_l g_l^\lambda)^2}{2(\mathcal{K}_{ii} + \mathcal{K}_{ll} - 2\mathcal{K}_{il})}$;
  solve (15) to update $\lambda_{i,j}, \gamma_{i,j}$;
  update $A_i^{\gamma_i}, B_i^{\gamma_i}$, and $g_l, \forall l = 1, \ldots, m$;
until $\max_{i \in I_{\text{up}}^{\lambda,\gamma}} y_i g_i - \min_{j \in I_{\text{down}}^{\lambda,\gamma}} y_j g_j < e$;
Calculate $b := \frac{1}{2}\left(\max_{i \in I_{\text{up}}^{\lambda,\gamma}} y_i g_i + \min_{j \in I_{\text{down}}^{\lambda,\gamma}} y_j g_j\right)$.

4. Numerical Experiments

In the above sections, we gave the SMO algorithms for training pin-SVM (7) and sparse pin-SVM (14). In the following, we will evaluate their performance on real-life data sets. There are two concerned aspects. First, we will test whether SMO is effective for training pin-SVMs. Second, with an effective training method, we can consider more experiments and support the theoretical analysis in [9]. The sparsity of sparse pin-SVM is also considered.

The data in these experiments are downloaded from the UCI Repository of Machine Learning Datasets [23] and LIBSVM data sets [17]. For some of these data, the training and test sets are provided. Otherwise, we randomly select m observations to train the classifier and use the remaining for test. The problem dimension n, the number of the training data m, and the number of test data m_T are summarized in Table 1.

In pin-SVM (7), we use the RBF kernel and apply Algorithm 2 to train the classifiers with different τ values. With the data size m grows, caching for the kernel matrix becomes larger. In our experiments, when $m \geq 5000$, we calculate element $\mathcal{K}_{ij}$ only when needed, which reduces the caching but costs more time. To make a fair comparison,

Table 1: Dimension, Training Data and Test Data Size

name	n	m	m_T	name	n	m	m_T
Spect	21	80	187	Pima	8	500	269
Monk3	6	122	432	Breast	10	500	199
Monk1	6	124	432	Splice	60	500	2175
Haberman	3	150	156	Spambase	58	1000	3601
Statlog	13	150	120	Guide1	4	3000	4000
Monk2	6	169	432	Magic	10	10000	9020
Ionosphere	33	200	151	IJCNN1	22	20000	91707
Transfusion	4	300	448	Cod RNA	8	30000	271617

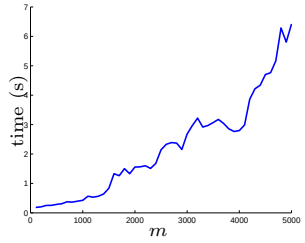
we use $\lambda_i = -\tau C_i$ as the initial solution. If the number of the training data is less than 10000, 10-fold cross-validation is utilized to tune the regularization coefficient C_0 and the bandwidth in the RBF kernel σ. Otherwise, we set $C_0 = 1$ and tune σ only. The training and test process is repeated 10 times. Then the average accuracy on test sets, the standard deviation, and the average computing time are reported in Table 2.

Table 2: Test Accuracy and Average Training Time

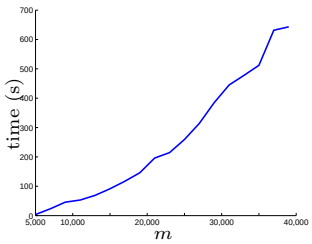
Data	$\tau = 0$	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$
Spect	**84.62** ± 3.22 8.96 ms	82.42 ± 2.47 9.06 ms	80.08 ± 1.73 8.92 ms	80.33 ± 2.63 8.94 ms
Monk3	92.22 ± 1.31 16.5 ms	93.80 ± 2.42 20.5 ms	**94.26** ± 1.36 25.5 ms	93.23 ± 1.46 28.6 ms
Monk1	81.97 ± 1.49 18.8 ms	82.31 ± 1.67 22.6 ms	83.06 ± 3.80 24.2 ms	**83.70** ± 4.12 27.0 ms
Haber.	74.27 ± 2.53 26.5 ms	73.63 ± 2.96 24.4 ms	72.61 ± 4.33 24.9 ms	**74.52** ± 2.63 24.6 ms
Statlog	82.82 ± 2.05 24.2 ms	**83.40** ± 2.01 27.8 ms	83.15 ± 1.42 30.2 ms	82.32 ± 2.39 32.8 ms
Monk2	83.98 ± 1.25 29.4 ms	85.56 ± 0.38 34.3 ms	**86.11** ± 0.00 37.1 ms	85.93 ± 0.28 39.4 ms
Iono.	94.01 ± 1.22 32.8 ms	**94.08** ± 1.39 40.6 ms	93.42 ± 1.16 44.4 ms	93.62 ± 1.32 47.4 ms
Trans.	73.64 ± 2.42 34.5 ms	73.70 ± 2.02 28.4 ms	73.74 ± 2.57 29.0 ms	**74.15** ± 2.28 28.8 ms
Pima	74.14 ± 2.45 111 ms	**74.51** ± 3.39 126 ms	73.77 ± 2.54 135 ms	73.36 ± 3.14 144 ms
Breast	**95.65** ± 1.42 57.4 ms	95.60 ± 1.66 71.3 ms	95.30 ± 2.02 73.9 ms	95.45 ± 1.69 74.0 ms
Splice	85.72 ± 3.34 102 ms	86.12 ± 0.70 93.2 ms	85.93 ± 1.18 98.9 ms	**86.25** ± 1.04 99.3 ms
Spamb.	**91.92** ± 0.38 200 ms	90.27 ± 0.47 168 ms	89.61 ± 1.07 171 ms	89.29 ± 2.02 167 ms
Guide1	**96.60** ± 0.21 158 ms	96.42 ± 0.28 181 ms	96.34 ± 0.16 195 ms	96.12 ± 0.79 210 ms
Magic	85.01 ± 0.24 23.4 s	**85.15** ± 0.48 29.0 s	84.31 ± 0.44 30.1 s	83.79 ± 0.69 30.3 s
IJCNN1	92.62 ± 0.65 147 s	**93.75** ± 1.13 212 s	92.42 ± 0.91 213 s	92.12 ± 1.11 209 s
RNA	**94.26** ± 1.05 123 s	92.26 ± 0.95 114 s	92.11 ± 1.11 124 s	91.08 ± 0.89 141 s

We also illustrate the scalability of the proposed SMO algorithm by plotting the training time for different training data sizes. In Fig.3 we plot the training time for data set IJCNN1. Notice that there is a sudden change at $m = 5000$, due to different kernel computation strategies.

Both Table 2 and Fig.3 illustrate that the proposed SMO method can train pin-SVM effectively. For different τ values, the computational time is similar and is not monotonic with respect to τ. In our method, pin-SVM is trained in the dual space, which corresponds to a QP with box constraints $-\tau C_i \le \lambda_i \le C_i$. One can observe that τ controls the size of the feasible set. In two extreme cases, i.e., when the box is large enough or very small, optimal solutions can be obtained easily. Therefore, though a larger τ is generally related to more training time, the difference is not significant. In some applications, a larger τ even corresponds to less training time. Generally, the proposed



(a)



(b)

Figure 3: Training time of Algorithm 2 ($\tau = 0.1$) for IJCNN1 for different training data sizes. (a) $m < 5000$; (b) $m \ge 5000$.

SMO for pin-SVM is effective and it takes similar training time as SMO for C-SVM.

With a properly selected τ, pin-SVM provides better classification accuracy over C-SVM. But the sparseness is lost. If the problem size is not too large and sparseness is not the main target, then finding a suitable τ is meaningful for improving the classification accuracy. Moreover, we can use sparse pin-SVM (14) to enhance the sparsity. In the following, we set $\tau = 0.1$ and apply Algorithm 3 for several different ε values. The training and test process is similar to the previous experiment, except that the parameters for sparse pin-SVM are tuned based on pin-SVM, since Algorithm 3 costs more time than Algorithm 2. In practice, if the allowed time is not strict, one can tune the parameters based on sparse pin-SVM and improve the performance further. The average classification accuracy, the number of support vectors (in brackets), and the training time are reported in Table 3, where the results of C-SVM are given as well for reference.

Compared with pin-SVM (7), sparse pin-SVM (14) enhances the sparsity, but takes more training time. In Algorithm 3, the update formulation involves a 4-dimensional QP problem. Though it can be solved effectively, its computation time is larger than the explicit update formulation in Algorithm 2. Roughly, Algorithm 3 needs 10 times more than Algorithm 2. In C-SVM, the points with $y_i f(x_i) < 1$ are related to zero dual variables and so are the points with $-\frac{\varepsilon}{\tau} < y_i f(x_i) < \varepsilon$ in sparse pin-SVM. Thus, the results of C-SVM are generally more sparse. But when the feature noise is heavy, it is worthy considering Algorithm 3 to train sparse pin-SVM.

Table 3: Test Accuracy, Number of Nonzero Dual Variables, and Training Time for Sparse pin-SVM ($\tau = 0.1$)

Data	C-SVM	$\varepsilon = 0.05$	$\varepsilon = 0.10$	$\varepsilon = 0.20$
Spect	84.62 (69)	82.20 (66)	80.80 (62)	79.50 (60)
	8.96 ms	108 ms	75.3 ms	90.4 ms
Monk3	92.22 (83)	90.58 (97)	91.44 (87)	90.42 (86)
	16.5 ms	143 ms	131 ms	127 ms
Monk1	81.97 (68)	79.15 (100)	77.53 (93)	74.84 (88)
	18.8 ms	126 ms	139 ms	127 ms
Haber.	74.27 (140)	74.63 (140)	74.63 (139)	73.63 (137)
	26.5 ms	154 ms	150 ms	177 ms
Statlog	82.82 (99)	83.11 (122)	82.11 (118)	81.41 (101)
	24.2 ms	155 ms	143 ms	139 ms
Monk2	83.98 (101)	83.87 (107)	83.78 (98)	81.45 (90)
	29.4 ms	246 ms	277 ms	253 ms
Iono.	94.01 (99)	93.89 (109)	93.80 (98)	93.76 (87)
	32.8 ms	277 ms	243 ms	243 ms
Trans.	73.64 (286)	75.53 (272)	74.32 (261)	73.37 (195)
	34.5 ms	250 ms	252 ms	250 ms
Pima	74.14 (337)	74.01 (354)	71.29 (346)	70.74 (336)
	111 ms	535 ms	502 ms	486 ms
Breast	95.65 (89)	96.50 (137)	95.85 (126)	93.60 (99)
	57.4 ms	445 ms	469 ms	483 ms
Splice	85.72 (271)	83.11 (392)	82.87 (322)	82.49 (234)
	102 ms	749 ms	652 ms	659 ms
Spamb.	91.92 (290)	91.28 (906)	91.12 (864)	91.20 (780)
	200 ms	741 ms	755 ms	697 ms
Guide1	96.60 (345)	96.72 (2018)	96.63 (1684)	94.99 (1203)
	158 ms	2.74 s	2.53 s	2.34 s

5. Conclusion

In this paper, sequential minimal optimization has been established for the support vector machine with the pinball loss. Since pin-SVM has the same problem structure as C-SVM, the corresponding SMO is related to that for C-SVM. We investigated the details and implemented SMO for pin-SVM. The SMO for training sparse pin-SVM was given as well. Then the proposed algorithms were evaluated on numerical experiments, showing the effectiveness of training pin-SVMs. The proposed SMO algorithms make pin-SVMs promising tools in real-life application, especially when the data are corrupted by feature noise.

Acknowledgment

The authors would like to thank Prof. Chih-Jen Lin in National Taiwan University for encouraging us to establish the SMO algorithm for pin-SVM.

The authors are grateful to the anonymous reviewers for helpful comments.

References

[1] C. Cortes and V. Vapnik, Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[2] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[3] X. Zhang. Using class-center vectors to build support vector machines. In *Proceedings of the IEEE Signal Processing Society Workshop*, pages 3–11. IEEE, 1999.

[4] J. Bi and T. Zhang. Support vector classification with input data uncertainty. In *Advances in Neural Information Processing Systems*, volume 17, page 161. MIT Press, 2005.

[5] G.R.G. Lanckriet, L.E. Ghaoui, C. Bhattacharyya, and M.I. Jordan. A robust minimax approach to classification. *The Journal of Machine Learning Research*, 3:555–582, 2003.

[6] P.K. Shivaswamy, C. Bhattacharyya, and A.J. Smola. Second order cone programming approaches for handling missing and uncertain data. *The Journal of Machine Learning Research*, 7:1283–1314, 2006.

[7] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009.

[8] B. Schölkopf, A.J. Smola, R.C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5): 1207–1245, 2000.

[9] X. Huang, L. Shi, and J.A.K. Suykens. Support vector machine classifier with pinball loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5): 984–997, 2014.

[10] R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

[11] I. Steinwart and A. Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1): 211–225, 2011.

[12] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods – Support Vector Learning*, pages 185–208. MIT Press, 1999.

[13] R.E. Fan, P.H. Chen, and C.J. Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

[14] L. Bottou and C.-J. Lin. Support vector machine solvers. in *Large Scale Kernel machines*, pages 301–320. MIT Press, 2007.

[15] Y. Torii and S. Abe. Decomposition techniques for training linear programming support vector machines. *Neurocomputing*, 72(4):973–984, 2009.

[16] J. Shawe-Taylor and S. Sun. A review of optimization methodologies in support vector machines. *Neurocomputing*, 74(17):3609–3618, 2011.

[17] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[18] C.C. Chang, C.W. Hsu, and C.J. Lin. The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008, 2000.

[19] C.J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, 2001.

[20] S.S. Keerthi and E.G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46 (1-3):351–360, 2002.

[21] D. Hush, P. Kelly, C. Scovel, and I. Steinwart. QP algorithms with guaranteed accuracy and run time for support vector machines. *The Journal of Machine Learning Research*, 7:733–769, 2006.

[22] J. López and J.R. Dorronsoro. Simple proof of convergence of the SMO algorithm for different SVM variants. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1142–1147, 2012.

[23] A. Frank and A. Asuncion. UCI machine learning repository, 2010.