Neural Process Lett (2016) 43:887–903
DOI 10.1007/s11063-015-9456-z

Coordinate Descent Algorithm for Ramp Loss Linear Programming Support Vector Machines

Xiangming Xi[1] · Xiaolin Huang[2] · Johan A. K. Suykens[2] · Shuning Wang[1]

Published online: 10 July 2015
© Springer Science+Business Media New York 2015

Abstract In order to control the effects of outliers in training data and get sparse results, Huang et al. (J Mach Learn Res 15:2185–2211, 2014) proposed the ramp loss linear programming support vector machine. This combination of l_1 regularization and ramp loss does not only lead to the sparsity of parameters in decision functions, but also limits the effects of outliers with a maximal penalty. However, due to its non-convexity, the computational cost to achieve a satisfying solution is often expensive. In this paper, we propose a modified coordinate descent algorithm, which deals with a series of one-variable piecewise linear subproblems. Considering that the obtained subproblems are DC programming problems, we linearize the concave part of the objective functions and solve the obtained convex problems. To test the performances of the proposed algorithm, numerical experiments have been carried out and analysed on benchmark data sets. To enhance the sparsity and robustness, the experiments are initialized from C-SVM solutions. The results confirm its excellent performances in classification accuracy, robustness and efficiency in computation.

Keywords Support vector machines · Ramp loss · l_1 Regularization · Coordinate descent algorithm

✉ Xiangming Xi
xixiangming@gmail.com; xxm10@mails.tsinghua.edu.cn

Xiaolin Huang
xiaolin.huang@esat.kuleuven.be

Johan A. K. Suykens
johan.suykens@esat.kuleuven.be

Shuning Wang
swang@mail.tsinghua.edu.cn

1 Department of Automation, Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China

2 Department of Electrical Engineering, ESAT-STADIUS, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium

1 Introduction

Ever since the proposal of support vector machines (SVM) in [5] and [10], SVM has become one of the most useful tools in classification. For a classification problem, our purpose is to find an optimal decision function to separate the training data, say $\{(\mathbf{x_i}, y_i)\}_{i=1}^m$, where $\mathbf{x_i} \in \mathbb{R}^n$ is a sample point and $y_i \in \{-1, +1\}$ is the corresponding class label. A generalized formulation for SVM can be written as follows,

$$\min_{\zeta, \mathrm{g}, b} \mu ||\mathrm{g}||_{\mathrm{K}}^2 + \frac{1}{m} \sum_{i=1}^{m} \mathrm{L}(\zeta_i) \tag{1a}$$

$$\text{s.t. } y_i(\mathrm{g}(\mathbf{x}_i) + b) \geqslant 1 - \zeta_i, i = 1, \ldots, m, \tag{1b}$$

$$\zeta \geqslant 0, \tag{1c}$$

where g belongs to the Reproducing Kernel Hilbert Space induced by the Mercer kernel K with norm $||\cdot||_{\mathrm{K}}$ [2], $\zeta \in \mathbb{R}^m$ are slack variables, $\mathrm{L}(\zeta_i)$ represents the loss function, $b \in \mathbb{R}$ is the offset, and μ is the regularization parameter. The objective (1a) is of great importance to the performance of the classification ability. The first term, $||\mathrm{g}||_{\mathrm{K}}^2$, is known as a regularization, of which the common forms include 1-norm (l_1 regularization [11]), 2-norm (l_2 regularization [10]), or sometimes the ∞-norm (l_∞ regularization [18]). And the commonly used loss functions include the hinge loss (L_1 loss [36]), squared hinge loss (L_2 loss [35]) and least square loss [30].

During the development of SVMs, much effort has been made on theoretical analysis of different combinations of regularization forms and loss functions in literature, and most discussions are concerning l_2 regularized SVMs. Chang et al. [8] propose a fast convergent algorithm based on a sufficient decreasing condition and a modified Newton method. Another technique they use to address the non twice differentiability of the squared hinge loss function is the generalized Hessian matrix proposed by Mangasarian [21]. Other discussions related to l_2 regularization can be found in [29] and [17] on non-parallel twin support vector machines and multi-class SVMs.

Besides, l_1 regularization attracts much attention for its sparsity. However, studies concerning l_1 regularization require the loss function to be differentiable or even twice differentiable, so that the mature gradient based techniques can be applied and convergence analysis could be conducted. Mangasarian [22] investigates the minimization of an l_1 regularized linear loss problem in the primal form. By obtaining the exact solution to an unconstrained squared hinge function, Fung and Mangasarian [12] propose a coordinate descent algorithm which exploits a generalized Newton method. Schmidt et al. [27] put forward two methods to address the non-differentiability of the l_1 regularized minimization with continuous and twice differentiable loss function, by smoothing and reformulating the problem as a non-negatively constrained optimization problem. Other discussions include [11,28,31,35], etc.

As the hinge loss, squared hinge loss and least square loss all enjoy convexity, they have been investigated by many researchers. However, when extreme outliers exist in training data, the choice of ramp loss will improve the robustness of SVMs against outliers and lead to better performances [4,25]. The first algorithm for binary SVM with ramp loss is proposed by Xu et al. [34], who utilize semi-definite programming relaxation. Wu and Liu present the theoretical analysis of the ramp loss SVM and investigate its advantages in sparsity and robustness in [33]. Later researchers reformulate SVM with ramp loss as mixed integer nonlinear programming problems, and then use a branch and bound method [19], heuristic algorithms [6] or commercial softwares [7] to obtain a solution. Another method proposed by

Collobert et al. in [9] is based on transforming the non-convex ramp loss SVM into a series of convex problems which can be solved by concave-convex procedure. Similar technique is utilized in [32] on the smooth ramp loss SVMs. In addition, Huang et al. [15] propose the ramp loss linear programming support vector machine (ramp-LPSVM). After investigating the formulation of ramp-LPSVM and its theoretical properties, they establish a global search algorithm, which is based on algorithms for DC problems [1], and Hill Detouring Method for concave optimization [16]. Though numerical results confirm that this algorithm outperforms traditional C-SVM in accuracy and robustness, the computational cost for a global search is more expensive. Even though, most of the algorithms mentioned above fail to solve large-scale problems, and could not guarantee the optimality. As interested in optimizing the problem with better performances in efficiency, we will discuss a fast convergent algorithm for ramp-LPSVM in this paper, i.e., a modified coordinate descent algorithm (CDA). Besides the analysis on convergence, its performances on accuracy and robustness are verified by numerical experiments.

The remainder of this paper is organized as follows. Section 2 describes the general concept of coordinate descent algorithm, and then we propose the algorithm for ramp-LPSVM, together with the formulation and convergence analysis of subproblems. The implementation issues are discussed in Sect. 3. Experiments concerning the comparison of CDA with other algorithms can be found in Sect. 4, while Sect. 5 concludes this paper.

2 Coordinate Descent Algorithm for Ramp-LPSVM

2.1 Coordinate Descent Algorithm

The coordinate descent algorithm is an efficient method for multi-variate optimization. It originates from the research of Hildreth [13], which solves unconstrained quadratic programming problems. Unlike other methods, the main concept of CDA is to optimize one single entry of the independent variables each time, so that the original problem is reduced to a series of one-variable subproblems. If they are convex and differentiable, the subproblems can be efficiently solved with mature techniques. After each entry is optimized iteratively, the algorithm eventually converges under the given assumption of continuity and differentiability [20]. The general framework of CDA is demonstrated in Algorithm 1.

Algorithm 1. General framework of coordinate descent algorithm.

Initialization
- Set algorithm parameters;
- Give an initial solution;

Outer Iteration: repeat
- **Inner Iteration: for** $i = 1, \ldots, m$ **do**
  - ◇ Express and solve the subproblem;
  - ◇ **if** *Gradient is not zero* **then**
    - ;
      - ○ Update the corresponding entry with the newly obtained optimum of subproblem;
- **end**

until *Termination condition is satisfied.*;

The coordinate descent algorithm is first introduced to support vector machines by Mangasarian and Musicant [23]. As the problem discussed refers to l_2 regularized problems, the

optimization method performs well. Since then, researchers have extended its application to more complex SVM formulations, such as l_2 regularized SVMs with hinge loss [14], squared hinge loss [8], least square loss [11], and l_1 regularized problems with logistic loss [35] and so on.

Most of the previous studies concerning coordinate descent algorithm focus on optimizing differentiable SVM problems. Generally speaking, the main computational cost is up to the complexity of the method for solving subproblems, and for a general problem, the cost is often so expensive that it makes CDA applicable to small or medium scaled problems. In order to improve the efficiency of CDA, Chang et al. [8] and Hsieh et al. [14] propose some useful tricks to achieve fast convergence of their algorithms for large-scale SVM problems with linear mapping of the training data. Even though, few attentions are paid on solving l_1 regularized SVMs with non-differentiable losses in literature. Considering that ramp-LPSVM has good performances in robustness and sparsity, we are interested in proposing a fast convergent coordinate descent algorithm.

As stated in [15], ramp-LPSVM can be formulated as follows,

$$\min_{\boldsymbol{\alpha}\geqslant 0, b} \ \mathrm{f}(\boldsymbol{\alpha}, b) = \mu \sum_{i=1}^{m} \alpha_i + \frac{1}{m} \sum_{i=1}^{m} \mathrm{L}_{\mathrm{ramp}}(1 - y_i(\mathrm{g}(\boldsymbol{\alpha}, \mathbf{x_i}) + b)), \tag{2}$$

where $\mathrm{L}_{\mathrm{ramp}}(u) = \max\{u, 0\} - \max\{u - 1, 0\}$ is the loss function, $\mathrm{g}(\boldsymbol{\alpha}, \mathbf{x}) = \sum_{j=1}^{m} \alpha_j y_j \mathrm{K}(\mathbf{x}, \mathbf{x_j})$ is the projection function, $\mathrm{K}(\cdot)$ is the kernel function and b is the offset.

The non-convexity and non-concavity of $\mathrm{L}_{\mathrm{ramp}}(u)$ make the standard coordinate descent algorithm fail to guarantee the optimality of the final solution. Denote $\boldsymbol{\xi} = (\boldsymbol{\alpha}^T, b)^T$, then we provide the compact expression of problem (2) as follows.

$$\begin{aligned} \min_{\boldsymbol{\xi}} \ & \mathrm{f}(\boldsymbol{\xi}) = \mathbf{e_0}^T \boldsymbol{\xi} + \frac{1}{m} \sum_{i=1}^{m} \max\left\{1 + \mathbf{c_i}^T \boldsymbol{\xi}, 0\right\} - \frac{1}{m} \sum_{i=1}^{m} \max\left\{\mathbf{c}_i^T \boldsymbol{\xi}, 0\right\} \\ \text{s.t.} \ & \xi_i \geqslant 0, i = 1, \ldots, m, \end{aligned} \tag{3}$$

where $\mathbf{e_0} = (\underbrace{\mu, \ldots, \mu}_{m}, 0)^T$, $\mathbf{C} = (\mathbf{c_1}^T, \mathbf{c_2}^T, \cdots, \mathbf{c_m}^T)$ with

$$c_{ij} = \begin{cases} -y_i y_j \mathrm{K}(\mathbf{x_i}, \mathbf{x_j}), & j = 1, \ldots, m \\ -y_i, & j = m + 1 \end{cases}, i = 1, \ldots, m. \tag{4}$$

Denote the variable to be optimized by $\boldsymbol{\xi}_{k,j} \in \mathbb{R}^m$, where the first subscript "k" represents the kth outer iteration in Algorithm 1, and the second "j" stands for the jth entry. Denote the obtained solution after the kth outer iteration by $\boldsymbol{\xi}_{k+1}$. To minimize (3) over the jth entry of $\boldsymbol{\xi}_{k,j}$, denoted by $\boldsymbol{\xi}_{k,j}^{(j)}$, we express the objective in the following way,

$$\begin{aligned} & \mathrm{h}(z; \boldsymbol{\xi}_{k,j}) \\ & = \mathrm{f}(\boldsymbol{\xi}_{k,j} + \mathbf{e_j}(z - \boldsymbol{\xi}_{k,j}^{(j)})) \\ & = \mathbf{e_0}^T(\boldsymbol{\xi}_{k,j} + \mathbf{e_j}(z - \boldsymbol{\xi}_{k,j}^{(j)})) + \sum_{i=1}^{m} \max\left\{1 + \mathbf{c}_i^T \boldsymbol{\xi}_{k,j} + \mathbf{c_i}^T \mathbf{e_j}(z - \boldsymbol{\xi}_{k,j}^{(j)}), 0\right\} \Big/ m \\ & \quad - \sum_{i=1}^{m} \max\left\{\mathbf{c}_i^T \boldsymbol{\xi}_{k,j} + \mathbf{c_i}^T \mathbf{e_j}(z - \boldsymbol{\xi}_{k,j}^{(j)}), 0\right\} \Big/ m \end{aligned}$$

$$= \mathbf{e_0}^T \mathbf{e_j} z + \mathbf{e_0}^T \xi_{k,j} - \mathbf{e_0}^T \mathbf{e_j} \xi_{k,j}^{(j)} + \left(\sum_{i=1}^{m} \max \left\{ \mathbf{c_i}^T \mathbf{e_j} z + 1 + \mathbf{c}_i^T \xi_{k,j} - \mathbf{c_i}^T \mathbf{e_j} \xi_{k,j}^{(j)}, 0 \right\} \right.$$
$$\left. - \sum_{i=1}^{m} \max \left\{ \mathbf{c_i}^T \mathbf{e_j} z + \mathbf{c}_i^T \xi_{k,j} - \mathbf{c_i}^T \mathbf{e_j} \xi_{k,j}^{(j)}, 0 \right\} \right) \Big/ m \tag{5}$$

So the corresponding subproblem can be written as,

$$\min_{z \geqslant 0} \quad \mathrm{h}(z; \xi_{k,j}) \tag{6}$$

where the nonnegative constraint should be neglected when $j = m + 1$ (since $b \in \mathbb{R}$).

The outline of the proposed coordinate descent algorithm is presented in Algorithm 2. Since we are interested in reduction of objective values, the condition in Line 10 of Algorithm 2 ensures the monotonically decrease.

Algorithm 2. Coordinate descent algorithm for ramp-LPSVM.

1 **Initialization**
2 • Set accuracy parameters η; Calculate parameter matrix $\mathbf{C}$ according to (4);
3 • Set the initial solution ξ_0, and $k := 0$;
4 **Outer Iteration**:
5 **repeat**
6 **Inner Iteration**:
7 • **for** $j = 1$ **to** $m + 1$ **do**
8 ◇ Express and solve the corresponding subproblem (6). Record the optimum $z^{\star}_{k,j}$;
9 ◇ **if** $\mathrm{h}(z^{\star}_{k,j}; \xi_{k,j}) < \mathrm{h}(\xi^{(j)}_{k,j}; \xi_{k,j})$ **then**
10 ;
11 ○ Update $\xi_{k+1,j} := \xi_{k,j} + \mathbf{e_j}(z^{\star}_{k,j} - \xi^{(j)}_{k,j})$;
12 **end**
13 • $k := k + 1$;
14 **until** $||\xi_{k+1} - \xi_k|| \leqslant \eta$.;

3 Algorithm Implementation

The framework of the proposed coordinate descent algorithm is given in the previous section. More detailed implementation issues should be carefully considered so that the algorithm can work to its most. In this section, we will focus on the following issues: the minimization of subproblems, the choice of optimized coordinates and so on.

3.1 Solving Subproblems

Recall the subproblem (6). Unlike a general nonlinear optimization problem, the one-variable piecewise linearity makes it convenient to be solved. A simple method to obtain an optimum is to calculate and compare the objective values of all feasible non-differentiable points of $\mathrm{h}(z)$ as well as the bounds. It works well when the problem scale is not large, but as the problem size grows, the computational cost will significantly increase. In order to solve the subproblem more efficiently, we should reformulate the non-convex part of the objective function (5).

Considering that $\mathrm{h}(z;\xi_{k,j})$ is a DC function, we express it as

$$\mathrm{h}(z;\xi_{k,j}) = \bar{\mathrm{h}}(z;\xi_{k,j}) + \tilde{\mathrm{h}}(z;\xi_{k,j}),$$

i.e., a summation of a convex function $\bar{\mathrm{h}}(z;\xi_{k,j})$ and a concave function $\tilde{\mathrm{h}}(z;\xi_{k,j})$, where

$$\begin{aligned}\bar{\mathrm{h}}(z;\xi_{k,j}) &= \mathbf{e_0}^T\mathbf{e_j}z + \mathbf{e_0}^T\xi_{k,j} - \mathbf{e_0}^T\mathbf{e_j}\xi_{k,j}^{(j)} \\ &\quad + \sum_{i=1}^{m}\max\left\{\mathbf{c_i}^T\mathbf{e_j}z + 1 + \mathbf{c}_i^T\xi_{k,j} - \mathbf{c_i}^T\mathbf{e_j}\xi_{k,j}^{(j)}, 0\right\}\Big/m, \\ \tilde{\mathrm{h}}(z;\xi_{k,j}) &= -\sum_{i=1}^{m}\max\left\{\mathbf{c_i}^T\mathbf{e_j}z + \mathbf{c}_i^T\xi_{k,j} - \mathbf{c_i}^T\mathbf{e_j}\xi_{k,j}^{(j)}, 0\right\}\Big/m.\end{aligned}$$

The linearization of $\tilde{\mathrm{h}}(\cdot)$ will lead to a convex subproblem, which can be solved efficiently. Thus, we give the partial first-order linearization of $\mathrm{h}(z;\xi_{k,j})$ at z_0 as follows,

$$\hat{\mathrm{h}}(z;\xi_{k,j}) = \bar{\mathrm{h}}(z;\xi_{k,j}) + \tilde{\mathrm{h}}(z_0;\xi_{k,j}) + \tilde{\mathrm{h}}'(z_0;\xi_{k,j})(z - z_0), \tag{7}$$

where $\tilde{\mathrm{h}}'(z_0;\xi_{k,j})$ is the first order derivative of $\tilde{\mathrm{h}}$ over z at z_0. It is noticeable that $\tilde{\mathrm{h}}$ is not differentiable at some points, so we will adopt the subgradient method. For any $z \in \mathbb{R}$, the first order derivative of $\tilde{\mathrm{h}}(z;\xi_{k,j})$ is based on

$$\begin{aligned}\tilde{\mathrm{h}}'(z;\xi_{k,j})_+ &= -\left(\sum_{i\in S_1}\mathbf{c_i}^T\mathbf{e_j} + \sum_{i\in S_2}\max\{\mathbf{c_i}^T\mathbf{e_j}, 0\}\right)\Big/m, \\ \tilde{\mathrm{h}}'(z;\xi_{k,j})_- &= -\left(\sum_{i\in S_1}\mathbf{c_i}^T\mathbf{e_j} + \sum_{i\in S_2}\min\{\mathbf{c_i}^T\mathbf{e_j}, 0\}\right)\Big/m,\end{aligned} \tag{8}$$

where $S_1 = \{i \mid \mathbf{c_i}^T\mathbf{e_j}z + \mathbf{c}_i^T\xi_{k,j} - \mathbf{c_i}^T\mathbf{e_j}\xi_{k,j}^{(j)} > 0, i = 1,\ldots,m\}$, $S_2 = \{i \mid \mathbf{c_i}^T\mathbf{e_j}z + \mathbf{c}_i^T\xi_{k,j} - \mathbf{c_i}^T\mathbf{e_j}\xi_{k,j}^{(j)} = 0, i = 1,\ldots,m\}$. The subscript "+" and "−" indicate the right and left derivative, respectively. For the non-differentiable points, the subgradient is defined as $\tilde{\mathrm{h}}'(z;\xi_{k,j}) = \lambda\tilde{\mathrm{h}}'(z;\xi_{k,j})_+ + (1-\lambda)\tilde{\mathrm{h}}'(z;\xi_{k,j})_-$, where $\lambda \in [0,1]$.

In this way, the subproblem is transformed to a convex problem in $\mathbb{R}$ and the computational cost for an optimum is much less than that when solving problem (6). A useful property of the subgradient of $\hat{\mathrm{h}}$ is provided as follows, which will be exploited in the convergence analysis in the successive section.

Proposition 1 *For a convex function* $\hat{\mathrm{h}}$ *in the form of (7), the subgradient of* $\hat{\mathrm{h}}$ *at any point* $z \in \mathbb{R}$ *satisfies*

$$||\hat{\mathrm{h}}'(z;\xi_{k,j})|| \leqslant 4||C||_1/m + 1.$$

Moreover,

$$||\hat{\mathrm{h}}'(z;\xi_{k,j})\mid_{z=z_0}|| \leqslant ||C||_1/m + 1.$$

Proof From the expressions of (7) and (8), we can get the subgradient of $\hat{\mathrm{h}}$ at any point $z \in \mathbb{R}$ as follows,

$$\hat{\mathrm{h}}'(z;\xi_{k,j}) = \bar{\mathrm{h}}'(z;\xi_{k,j}) + \tilde{\mathrm{h}}'(z_0;\xi_{k,j}).$$

Besides the expression of $\tilde{\mathrm{h}}'$ in (8), the $\bar{\mathrm{h}}'(z;\xi_{k,j})$ is given as follows,

$$\bar{\mathrm{h}}'(z;\xi_{k,j})_+ = \mathbf{e_0}^T\mathbf{e_j} + \left(\sum_{i\in S_3}\mathbf{c_i}^T\mathbf{e_j} + \sum_{i\in S_4}\max\{\mathbf{c_i}^T\mathbf{e_j},0\}\right)\Big/m,$$

$$\bar{\mathrm{h}}'(z;\xi_{k,j})_- = \mathbf{e_0}^T\mathbf{e_j} + \left(\sum_{i\in S_3}\mathbf{c_i}^T\mathbf{e_j} + \sum_{i\in S_4}\min\{\mathbf{c_i}^T\mathbf{e_j},0\}\right)\Big/m,$$

where $S_3 = \{i \mid \mathbf{c_i}^T\mathbf{e_j}z + 1 + \mathbf{c}_i^T\xi_{k,j} - \mathbf{c_i}^T\mathbf{e_j}\xi_{k,j}^{(j)} > 0, i = 1,\ldots,m\}$, $S_4 = \{i \mid \mathbf{c_i}^T\mathbf{e_j}z + 1 + \mathbf{c}_i^T\xi_{k,j} - \mathbf{c_i}^T\mathbf{e_j}\xi_{k,j}^{(j)} = 0, i = 1,\ldots,m\}$. Then we have

$$\begin{aligned}
&\hat{\mathrm{h}}'(z;\xi_{k,j})\\
&= \mathbf{e_0}^T\mathbf{e_j} + \sum_{i\in S_3}\mathbf{c_i}^T\mathbf{e_j}/m - \sum_{i\in S_1}\mathbf{c_i}^T\mathbf{e_j}/m\\
&\quad + \left(\lambda_1\sum_{i\in S_4}\max\left\{\mathbf{c_i}^T\mathbf{e_j},0\right\} + (1-\lambda_1)\sum_{i\in S_4}\min\left\{\mathbf{c_i}^T\mathbf{e_j},0\right\}/m\right)\\
&\quad - \left(\lambda_2\sum_{i\in S_2}\min\left\{\mathbf{c_i}^T\mathbf{e_j},0\right\} + (1-\lambda_2)\sum_{i\in S_2}\max\left\{\mathbf{c_i}^T\mathbf{e_j},0\right\}/m\right).
\end{aligned}$$

Now we turn to the relation between $S_i, i = 1,\ldots,4$. First, the definition of S_i leads to that $S_1\cap S_2 = \emptyset$ and $S_3\cap S_4 = \emptyset$. Second, since S_1, S_2 are determined by z_0, and S_3, S_4 by z, if taking $z = z_0$, then $S_1\subset S_3$. By using these two facts, there holds that $(S_3 - S_1)\cap S_2\cap S_4 = \emptyset$, and that

$$\begin{aligned}
&||\hat{\mathrm{h}}'(z;\xi_{k,j})\mid_{z=z_0}||\\
&= ||\mathbf{e_0}^T\mathbf{e_j}|| + ||\sum_{i\in S_3-S_1}\mathbf{c_i}^T\mathbf{e_j}/m||\\
&\quad + \left|\left|\left(\lambda_1\sum_{i\in S_4}\max\left\{\mathbf{c_i}^T\mathbf{e_j},0\right\}/m + (1-\lambda_1)\sum_{i\in S_4}\min\{\mathbf{c_i}^T\mathbf{e_j},0\}/m\right)\right|\right|\\
&\quad + \left|\left|\left(\lambda_2\sum_{i\in S_2}\min\{\mathbf{c_i}^T\mathbf{e_j},0\}/m + (1-\lambda_2)\sum_{i\in S_2}\max\{\mathbf{c_i}^T\mathbf{e_j},0\}/m\right)\right|\right|\\
&\leqslant 1 + \sum_{i\in(S_3-S_1)\cup S_2\cup S_4}|\mathbf{c_i}^T\mathbf{e_j}|/m\\
&\leqslant 1 + ||C_1||/m.
\end{aligned}$$

If $z\neq z_0$, then based on the similar analysis above, we can see that

$$\begin{aligned}
&||\hat{\mathrm{h}}'(z;\xi_{k,j})||\\
&\leqslant 1 + ||\sum_{i\in S_3}\mathbf{c_i}^T\mathbf{e_j}/m|| + ||\sum_{i\in S_1}\mathbf{c_i}^T\mathbf{e_j}/m||\\
&\quad + \left|\left|\left(\lambda_1\sum_{i\in S_4}\max\{\mathbf{c_i}^T\mathbf{e_j},0\}/m + (1-\lambda_1)\sum_{i\in S_4}\min\{\mathbf{c_i}^T\mathbf{e_j},0\})/m\right)\right|\right|
\end{aligned}$$

$$+\left|\left|\left(\lambda_2 \sum_{i\in S_2} \min\{\mathbf{c_i}^T \mathbf{e_j}, 0\}/m + (1-\lambda_2) \sum_{i\in S_2} \max\{\mathbf{c_i}^T \mathbf{e_j}, 0\}/m\right)\right|\right|$$
$$\leqslant 1 + 4||C_1||/m.$$

The last step above is because that though S_1, S_3 may share common elements, all S_i are subsets of $\{1, \ldots, m\}$, which leads to the last step by proper relaxation. □

Algorithm 3. Minimization of a subproblem (5).

Initialize
- Given an initial solution $\xi_{k,j}$, $z_0 = \xi_{k,j}^{(j)}$ and $\beta \in (0, 1)$; Denote lower bounds of z by b_l, and the upper bound b_u; Set $p := 0$;

repeat
- **if** $\xi_{k,j}^{(j)} = b_l$ **then**
  - Calculate $\hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)\,|_{z=\xi_{k,j}^{(j)+}}$;
  - **if** $\hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)\,|_{z=\xi_{k,j}^{(j)+}} \geqslant 0$ **then**
    - Let $z_{p+1} := z_p$;
  - **else**
    - Let $\gamma = b_u - b_l$, and do backtracking line search along $d = 1$. Stop the search when $\hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)\,|_{z=b_l+\gamma\beta^{s_{k,j}-1}} \geqslant 0$. Let $z_{p+1} := b_l + \gamma\beta^{s_{k,j}-1}$;
  - **end**
- **else if** $\xi_{k,j}^{(j)} = b_u$ **then**
  - Calculate $\hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)\,|_{z=\xi_{k,j}^{(j)-}}$;
  - **if** $\hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)\,|_{z=\xi_{k,j}^{(j)-}} < 0$ **then**
    - Let $z_{p+1} := z_p$;
  - **else**
    - Let $\gamma = b_u - b_l$, and do backtracking line search along $d = -1$. Stop the search when $\hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)\,|_{z=b_u-\gamma\beta^{s_{k,j}-1}} \leqslant 0$. Let $z_{p+1} := b_l + \gamma\beta^{s_{k,j}-1}$;
  - **end**
- **else**
  - Calculate $\hat{\mathrm{h}}'(b_l; \xi_{k,j}, z_p)\,|_{z=\xi_{k,j}^{(j)-}}$ and $\hat{\mathrm{h}}'(b_l; \xi_{k,j}, z_p)\,|_{z=\xi_{k,j}^{(j)+}}$;
  - **if** $0 \in \hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)$ **then**
    - Let $z_{p+1} := b_m$;
  - **else**
    - Let $\gamma = b_u - \xi_{k,j}^{(j)}$, and do backtracking line search along $d = 1$. (or $\gamma = \xi_{k,j}^{(j)} - b_l$, $d = -1$.) Stop the search when $\hat{\mathrm{h}}'(z; \xi_{k,j}, z_p)\,|_{z=b_l+\gamma\beta^{s_{k,j}-1}} \geqslant (\leqslant) 0$. Let $z_{p+1} := \xi_{k,j}^{(j)} + \gamma\beta^{s_{k,j}-1}$ ($z_{p+1} := \xi_{k,j}^{(j)} - \gamma\beta^{s_{k,j}-1}$);
  - **end**
- **end**
- $p := p + 1$;

until $|z_p - z_{p-1}| < \eta$;

As interested in finding the solution which satisfies $\hat{\mathrm{h}}'(z; \xi_{k,j}) = 0$, we will implement the detailed method with gradient (or subgradient) information of the linearized problem, which is illustrated in Algorithm 3.

Algorithm 3 exploits the fact that due to the convexity of $\hat{\mathrm{h}}$, the possible cases of derivatives at lower and upper bounds are *(non-negative, non-negative), (non-positive, non-positive), (non-positive, non-negative)*. In the first two cases, the optimum is the lower bound and upper bound, respectively, while only in the third case, an iterative searching procedure should be conducted to locate the optimum. Another issue we should notice is that for ramp-LPSVM, the value of the second term of f in problem (2) is within the interval [0, 1], so ξ should be bounded by a considerably small positive value. As a result, it is reasonable to set a fixed upper bound, say b_u, during the searching procedure, while to keep the optimum unchanged.

3.2 Convergence Analysis

Theorem 1 *Let* $\{\xi_k\}$, $\{\mathrm{f}(\xi_k)\}$ *be sequences generated by Algorithm 2, then we have the following convergence results.*

(1) $\{\mathrm{f}(\xi_k)\}$ *is nonincreasing and it satisfies*

$$\mathrm{f}(\xi_{k+1}) - \mathrm{f}(\xi_k) \leqslant -\sum_{j\in\hat{S}_k} \gamma\beta^{s_{k,j}-1}, \tag{9}$$

where $\hat{S}_k = \{j \mid z^\star_{k,j} \neq \xi^{(j)}_{k,j}\}$;

(2) If $\{\xi_k\}_{\mathcal{K}}$ *is a convergent subsequence of* $\{\xi_k\}$, *then* $\hat{S}_k \to \emptyset$, *and* $\{\mathrm{f}(\xi_{k+1}) - \mathrm{f}(\xi_k)\} \to 0$;

Proof (1) Take the j-th inner iteration of the k-th outer iteration into consideration, and the problem is

$$\begin{aligned} &\min_{z} \ \hat{\mathrm{h}}(z;\, \xi_{k,j}) \\ &s.t.\ \ z \in [b_l, b_u]. \end{aligned} \tag{10}$$

And due to the concavity of $\tilde{\mathrm{h}}$, there is

$$\begin{aligned} &\mathrm{h}(z^\star_{k,j};\, \xi_{k,j}) - \mathrm{h}(z_0;\, \xi_{k,j}) \\ &= \bar{\mathrm{h}}(z^\star_{k,j};\, \xi_{k,j}) + \tilde{\mathrm{h}}(z^\star_{k,j};\, \xi_{k,j}) - (\bar{\mathrm{h}}(z_0;\, \xi_{k,j}) + \tilde{\mathrm{h}}(z_0;\, \xi_{k,j})) \\ &\leqslant \hat{\mathrm{h}}(z^\star_{k,j};\, \xi_{k,j}) - \hat{\mathrm{h}}(z_0;\, \xi_{k,j}). \end{aligned} \tag{11}$$

Noticed that if $z_0 = \xi^{(j)}_{k,j}$, then we have $\mathrm{h}(z^\star_{k,j-1};\, \xi_{k,j-1}) = \mathrm{h}(z_0;\, \xi_{k,j})$. This leads to

$$\mathrm{h}\left(z^\star_{k,j};\, \xi_{k,j}\right) - \mathrm{h}(z^\star_{k,j-1};\, \xi_{k,j-1}) \leqslant \hat{\mathrm{h}}\left(z^\star_{k,j};\, \xi_{k,j}\right) - \hat{\mathrm{h}}\left(z^\star_{k,j-1};\, \xi_{k,j-1}\right).$$

Sum up the above inequalities over $j = 1, \ldots, n$, then we have

$$\mathrm{h}(z^\star_{k,n};\, \xi_{k,n}) - \mathrm{h}(z_0;\, \xi_{k,0}) \leqslant \sum_{j=1}^{n} \left(\hat{\mathrm{h}}\left(z^\star_{k,j};\, \xi_{k,j}\right) - \hat{\mathrm{h}}\left(\xi^{(j)}_{k,j};\, \xi_{k,j}\right)\right), \tag{12}$$

which is

$$\mathrm{f}(\xi_{k+1}) - \mathrm{f}(\xi_k) \leqslant \sum_{j=1}^{n} \left(\hat{\mathrm{h}}\left(z^\star_{k,j};\, \xi_{k,j}\right) - \hat{\mathrm{h}}\left(\xi^{(j)}_{k,j};\, \xi_{k,j}\right)\right). \tag{13}$$

Denote $\hat{S}_k = \{j \mid z^\star_{k,j} \neq \xi^{(j)}_{k,j}\}$, and $\hat{S}^c_k$ is its complementary set. Since for any $j \in \hat{S}^c_k$, it holds that $z^\star_{k,j} = \xi^{(j)}_{k,j}$ and $\hat{\mathrm{h}}'(z;\, \xi_{k,j})\,|_{z=\xi^{(j)}_{k,j}}\,(z^\star_{k,j} - \xi^{(j)}_{k,j}) = 0$. Then (13) can be

expressed as

$$\begin{aligned} \mathrm{f}(\xi_{k+1}) - \mathrm{f}(\xi_k) &\leqslant \sum_{j=1}^{n} \left(\hat{\mathrm{h}}(z^{\star}_{k,j}; \xi_{k,j}) - \hat{\mathrm{h}}(0; \xi_{k,j}) \right) \\ &\leqslant \sum_{j \in \hat{S}_k} \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi^{(j)}_{k,j}} \left(z^{\star}_{k,j} - \xi^{(j)}_{k,j} \right). \end{aligned} \tag{14}$$

However, as $\hat{\mathrm{h}}$ is not differentiable at some points, we need appeal to the subgradient of $\hat{\mathrm{h}}$. Remind the expression of $\hat{\mathrm{h}}$ in (7), and its subgradient from above and from below is defined as

$$\begin{aligned} \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)-}_{k,j}} &= \bar{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)-}_{k,j}} - \tilde{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi^{(j)}_{k,j}}, \\ \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)+}_{k,j}} &= \bar{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)+}_{k,j}} - \tilde{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi^{(j)}_{k,j}}, \end{aligned} \tag{15}$$

where $\tilde{\mathrm{h}}'(z; \xi_{k,j}) \mid^{(j)}_{z=\xi^{(j)}_{k,j}}$ is the subgradient of $\tilde{\mathrm{h}}(z; \xi_{k,j})$ at $\xi_{k,j}$. According to Algorithm 3, if $z^{\star}_{k,j} \neq \xi_{k,j}$, one of the following conditions holds,

- $\xi^{(j)}_{k,j} \in (b_l, b_u)$, $\hat{\mathrm{h}}'_{-}(z; \xi_{k,j}) \mid_{z \to \xi^{(j)-}_{k,j}} > 0$, and $\hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)+}_{k,j}} > 0$;
- $\xi^{(j)}_{k,j} \in (b_l, b_u)$, $\hat{\mathrm{h}}'_{-}(z; \xi_{k,j}) \mid_{z \to \xi^{(j)-}_{k,j}} < 0$, and $\hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)+}_{k,j}} < 0$;
- $\xi^{(j)}_{k,j} = b_l$, $\hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)+}_{k,j}} < 0$;
- $\xi^{(j)}_{k,j} = b_u$, $\hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z \to \xi^{(j)-}_{k,j}} > 0$.

In the first and fourth cases, as $\hat{\mathrm{h}}$ is convex, and the subgradient is a convex combination of the limitation from above and below, the optimum to problem (10), $z^{\star}_{k,j}$, must be smaller than $\xi^{(j)}_{k,j}$. While in the other cases, $z^{\star}_{k,j} > \xi^{(j)}_{k,j}$.

Define $\Delta_k \triangleq -\sum_{j \in \hat{S}_k} \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi^{(j)}_{k,j}} (z^{\star}_{k,j} - \xi^{(j)}_{k,j})$. Using the Armijo rules to search for $z^{\star}_{k,j}$ will lead to

$$\Delta_k = -\sum_{j \in \hat{S}_k} \gamma \beta^{s_{k,j}-1}, \tag{16}$$

where $d_{k,j} = sign(\hat{\mathrm{h}}'(\xi^{(j)}_{k,j}; \xi_{k,j}))$. Then

$$\mathrm{f}(\xi_{k+1}) - \mathrm{f}(\xi_k) \leqslant -\sum_{j \in \hat{S}_k} \gamma \beta^{s_{k,j}-1} \leqslant 0. \tag{17}$$

(2) Let $\{\xi_k\}_{\mathcal{K}}$ be a subsequence of $\{\xi_k\}$ which converges to $\bar{\xi}$. Then since $\{\mathrm{f}(\xi_k)\}$ is non-increasing and is lower bounded, it holds that $\mathrm{f}(\bar{\xi}) \leqslant \liminf_{k \to \infty, k \in \mathcal{K}} \mathrm{f}(\xi_k)$. Therefore, it implies that $\{\mathrm{f}(\xi)\}_{\mathcal{K}} \to \mathrm{f}(\bar{\xi})$, and thus, $\{\mathrm{f}(\xi_{k+1}) - \mathrm{f}(\xi_k)\} \to 0$. And then we have

$$\{\Delta_k\} \to 0.$$

Suppose that $\{\hat{S}_k\}_{\mathcal{K}}$ is not convergent to $\emptyset$. By a proper selection of $\mathcal{K}$, there exists $j_k \in \hat{S}_k$, for any $k \geqslant \bar{k}$, where $\bar{k}$ is a positive integer. Then after one outer iteration, the minimization of $\hat{\mathrm{h}}(z; \xi_{k,j})$ is solved with one of the following conditions,

(a) $\xi_{k,j_k}^{(j_k)} \in (b_l, b_u), 0 \notin \hat{\mathrm{h}}'(z; \xi_{k,j_k}) \mid_{z=\xi_{k,j_k}^{(j_k)}}$;

(b) $\xi_{k,j_k}^{(j_k)} = b_l, \hat{\mathrm{h}}'(z; \xi_{k,j_k}) \mid_{z=b_l^+} < 0$;

(c) $\xi_{k,j_k}^{(j_k)} = b_u, \hat{\mathrm{h}}'(z; \xi_{k,j_k}) \mid_{z=b_u^-} > 0$.

Moreover, there exists $\epsilon > 0$, such that $\Delta_k = \sum_{j \in \hat{S}_k} \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi_{k,j}^{(j)}} \left(z_{k,j}^{\star} - \xi_{k,j}^{(j)}\right) \leqslant \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi_{k,j_k}^{(j_k)}} \left(z_{k,j_k}^{\star} - \xi_{k,j_k}^{(j_k)}\right) \leqslant -\epsilon < 0$.

By summing up Eq. (14) for $k \in \mathcal{K}$, we have

$$\begin{aligned} \mathrm{f}(\xi_k) - \mathrm{f}(\xi_1) &\leqslant \sum_{j \in \hat{S}_k} \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi_{k,j}^{(j)}} (z_{k,j}^{\star} - \xi_{k,j}^{(j)}) \\ &\leqslant \hat{\mathrm{h}}'(z; \xi_{k,j}) \mid_{z=\xi_{k,j_k}^{(j_k)}} (z_{k,j_k}^{\star} - \xi_{k,j_k}^{(j_k)}) \\ &< k\epsilon, \end{aligned} \tag{18}$$

which implies

$$\inf \lim_{k \to \infty, k \in \mathcal{K}} \mathrm{f}(\xi_k) - \mathrm{f}(\xi_1) = -\infty. \tag{19}$$

A clear contradiction against that $\mathrm{f}(\xi)$ is lower bounded can be seen. Thus, $\{\hat{S}_k\} \to \emptyset$, and $\{\Delta_k\} \to 0$. □

Theorem 2 *Let* $\{\xi_k\}$, $\{\mathrm{f}(\xi_k)\}$ *be sequences generated by Algorithm 2. If the coefficient matrix* C *is proper, i.e.,* $||C||_1 < +\infty$, *then* $\{\mathrm{f}(\xi_k)\}$ *converges Q-linearly over the outer iterations.*

Proof Suppose that there exists a subsequence of $\{\xi_k\}$, denoted by $\{\xi_k\}_{\mathcal{K}}$, which converges to a point $\bar{\xi}$. The condition $||C||_1 < +\infty$ implies that the subgradient of f at any point is proper, and for the problem we discuss, it holds according to Proposition 1. Thus, we have

$$\mathrm{f}(\xi_k) - \mathrm{f}(\bar{\xi}) = \nabla \mathrm{f}(\tilde{\xi})^T (\xi_k - \bar{\xi}) \leqslant L||\xi_k - \bar{\xi}||, \tag{20}$$

where $||\nabla \mathrm{f}|| < L$ and $0 < L < \infty$. Moreover, there holds

$$\begin{aligned} ||\xi_k - \bar{\xi}|| &\leqslant ||\xi_{k+1} - \xi_k|| + ||\xi_{k+1} - \bar{\xi}|| \\ &\cdots \\ &\leqslant \lim_{N \to \infty} \sum_{l=1}^{N} ||\xi_{k+l+1} - \xi_{k+l}|| + ||\xi_{k+N+1} - \bar{\xi}|| \\ &\leqslant N_1 \sup_l ||\xi_{k+l+1} - \xi_{k+l}|| \\ &= N_1 \sup_l \left(\sum_{j \in \hat{S}_{k+l}} \left(\gamma \beta^{s_{k,j}-1} d_{k,j}\right)^2 \right)^{1/2}. \\ &= N_1 \sup_l \left(\sum_{j \in \hat{S}_{k+l}} \left(\gamma \beta^{s_{k,j}-1}\right)^2 \right)^{1/2} / \min_{j \in \hat{S}_{k+l}} |\hat{\mathrm{h}}' \left(\xi_{k,j}^{(j)}; \xi_{k,j}\right)| \\ &\leqslant N_2(-\Delta_k), \end{aligned} \tag{21}$$

where $\xi_{k+l} \in \{\xi_k\}_{\mathcal{K}}$, and $0 < N_2 < +\infty$. The third step of the above relation holds since for any points in $\{\xi_k\}_{\mathcal{K}}$, there exists $N_1 > 0$, such that for any $k > N_1$ and $\nu > 0$, $||\xi_k - \bar{\xi}|| < \nu$.

The last step is conducted following the fact that for any $j \in \hat{S}_k$, the subgradient of $\hat{\mathrm{h}}$ is nonzero, and thus, $\inf_k \min_{j \in \hat{S}_{k+l}} |\hat{\mathrm{h}}'(\xi_{k,j}^{(j)}; \xi_{k,j})| > 0$.

Combining the above two equations leads to

$$\begin{aligned} \mathrm{f}(\xi_k) - \mathrm{f}(\bar{\xi}) &\leqslant L||\xi_k - \bar{\xi}|| \\ &\leqslant LN_2(-\Delta_k) \\ &\leqslant LN_2(\mathrm{f}(\xi_{k+1}) - \mathrm{f}(\xi_k)). \end{aligned} \tag{22}$$

From (22), we can see that

$$\mathrm{f}(\xi_{k+1}) - \mathrm{f}(\bar{\xi}) \leqslant \frac{LN_2 - 1}{LN_2}(\mathrm{f}(\xi_k) - \mathrm{f}(\bar{\xi})), \tag{23}$$

which means $\{\mathrm{f}(\xi_k)\}$ converges to $\mathrm{f}(\bar{\xi})$ at least Q-linearly over outer iterations. □

Though f converges at least Q-linearly over outer iterations, Algorithm 2 might minimize every variables in each outer iteration, and this would take more computational time than that of the theoretical analysis. This is also verified by the numerical experiments in the following section.

3.3 Other Discussions

In literature, researchers have already discussed the influence of the updating order of coordinates on experimental performances of CDA. Shalev-Shwartz and Tewari [28] propose a statistic coordinate descent algorithm for l_1 regularized SVMs, and establish the corresponding convergence analysis. Later, an extension of the coordinate gradient descent algorithm is proposed in [31] to deal with SVM problems of which the objectives consist of smooth functions and separable convex functions. The l_2 regularized SVMs with several non-convex losses are discussed in [24]. When it comes to Algorithm 2, we will adopt the statistic order of the updated coordinates, while specifically, we choose the uniform distribution.

The proposed algorithm is sensitive to initial solutions due to the lack of convexity of f. Since l_1 regularization implies sparsity of SVMs, most of the entries of $\boldsymbol{\alpha}$ will be 0. A good trial is to set the initial solution of $\boldsymbol{\alpha}$ to be 0. The numerical experiment about the choice of initial solution can be found in Sect. 4.1. In the experiments, the randomness caused by the choice of coordinate orders will also be considered.

4 Numerical Experiments

In this section, we will verify the performance of the proposed algorithm, and compare it with C-SVM implemented via sequential minimal optimization [26]. The data set we used is selected in the UCI data library [3], including the IJCNN data set, SPECT heart data set and so on. We also compare the performances of different kernels, including the Gaussian kernel, $\mathrm{K}(\mathbf{x_i}, \mathbf{x_j}) = \exp(-||\mathbf{x_i} - \mathbf{x_j}||_2^2/\sigma^2)$, and the linear kernel, $\mathrm{K}(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$, and discuss the influence of different starting points. All the experiments are run on the Windows 7 platform of Matlab R2013a in Core(TM) i7-3770, 3.4GHz, 16 GB RAM. In order to overcome the limited memory issue, we utilize the kernel caching method in the bioinfo toolbox of Matlab for the data sets which contain more than 20,000 sample.

Table 1 Comparison of different strategies for starting points in average test accuracy (Acc.), number of support vectors (SV.) and training time (Time.) with the linear kernel

Data	m	Strategy	Alg. 2		
			Acc. (%)	SV.	Time. (s)
Spect	100	A	91.98 ± 0.0008	#60	0.016
		B	91.98 ± 0.0000	#26	0.026
Brst.	100	A	96.23 ± 0.0107	#79	0.027
		B	95.20 ± 0.8941	#15	0.018
Hbrn.	200	A	68.87 ± 0.0010	#83	0.030
		B	70.75 ± 0.0000	#106	0.035
IJCNN.	1000	A	80.86 ± 0.0000	#547	0.247
		B	80.86 ± 0.0004	#215	1.126
IJCNN.	4959	A	82.18 ± 0.0005	#2041	3.196
		B	86.56 ± 2.1802	#913	4.621
IJCNN.	9989	A	89.07 ± 0.9587	#1699	15.65
		B	87.72 ± 2.9350	#1677	17.07
IJCNN.	19, 992	A	90.24 ± 0.0719	#5010	48.95
		B	90.81 ± 0.9434	#3415	47.18
IJCNN.	49, 999	A	90.92 ± 1.1502	#8395	365.2
		B	91.17 ± 0.5099	#8030	373.1

Table 2 Comparison of different strategies for starting points in average test accuracy (Acc.), number of support vectors (SV.) and training time (Time.) with the Gaussian kernel

Data	m	Strategy	Alg. 2		
			Acc. (%)	SV.	Time. (s)
Spect	100	A	91.98 ± 0.0000	#44	0.016
		B	91.98 ± 0.0000	#28	0.021
Brst.	100	A	96.67 ± 0.8879	#63	0.016
		B	94.80 ± 0.0839	#21	0.022
Hbrn.	200	A	71.70 ± 0.0000	#119	0.035
		B	75.47 ± 0.0000	#111	0.023
IJCNN.	1000	A	80.86 ± 0.0000	#728	0.245
		B	80.87 ± 0.0001	#556	0.264
IJCNN.	4959	A	83.15 ± 0.0005	#2091	3.405
		B	87.34 ± 2.0605	#909	6.089
IJCNN.	9989	A	88.44 ± 0.0439	#6994	13.06
		B	88.33 ± 1.5710	#1697	14.94
IJCNN.	19, 992	A	90.20 ± 0.0957	#9918	62.50
		B	91.81 ± 3.1322	#3259	67.18
IJCNN.	49, 999	A	92.88 ± 1.3625	#9010	488.2
		B	93.10 ± 0.0008	#8122	425.3

4.1 Starting Solutions

As is discussed in Sect. 3.3, due to the non-convexity of ramp-LPSVM, different starting points will result in different classifiers. Considering the sparsity of the final solution, which

Table 3 Comparisons of C-SVM and the proposed coordinate descent algorithm with linear kernel in average test accuracy (Acc.), number of support vectors (SV.) and training time (Time.)

Data	m	r	C-SVM			Alg. 2		
			Acc. (%)	SV.	Time. (s)	Acc. (%)	SV.	Time. (s)
Spect	100	0.00	91.98 ± 0.0000	#67	0.031	91.98 ± 0.0000	#26	0.026
		0.05	91.66 ± 1.1878	#69	0.030	91.98 ± 0.0015	#27	0.017
		0.10	87.70 ± 0.4517	#70	0.030	91.98 ± 0.0008	#40	0.010
Brst.	100	0.00	93.55 ± 0.0007	#43	0.031	95.20 ± 0.8941	#15	0.018
		0.05	91.53 ± 0.0045	#50	0.028	95.40 ± 1.1490	#11	0.024
		0.10	90.69 ± 0.0098	#49	0.031	95.17 ± 0.4983	#12	0.022
Hbrn.	200	0.00	71.96 ± 2.0138	#110	0.085	70.75 ± 0.0000	#106	0.035
		0.05	73.65 ± 1.2311	#118	0.058	72.53 ± 1.7291	#108	0.058
		0.10	70.10 ± 1.3201	#126	0.060	69.22 ± 2.1584	#123	0.058
IJCNN.	1000	0.00	81.68 ± 0.0008	#312	1.186	80.86 ± 0.0004	#215	1.126
		0.05	78.09 ± 0.0008	#321	1.096	80.40 ± 0.0001	#244	1.190
		0.10	77.54 ± 0.0002	#299	1.364	79.94 ± 0.0002	#282	1.200
IJCNN.	4959	0.00	83.65 ± 1.5566	#864	10.03	86.56 ± 2.1802	#913	4.621
		0.05	80.18 ± 1.8731	#1032	11.25	86.52 ± 1.3450	#892	4.563
		0.10	79.36 ± 1.6542	#1158	10.21	84.82 ± 2.6946	#901	4.349
IJCNN.	9989	0.00	86.44 ± 0.0021	#2134	15.39	87.72 ± 2.9350	#1677	17.07
		0.05	81.41 ± 0.0006	#2208	16.81	86.38 ± 1.5506	#1697	15.53
		0.10	80.55 ± 0.0008	#2196	15.73	85.75 ± 1.0124	#1740	17.73
IJCNN.	19, 992	0.00	89.69 ± 2.1189	#4634	50.38	90.81 ± 0.9434	#3415	47.17
		0.05	87.21 ± 1.0080	#5102	51.02	87.50 ± 1.4738	#3306	48.55
		0.10	83.05 ± 1.0000	#5023	51.02	84.22 ± 0.3614	#3277	45.40
IJCNN.	49, 999	0.00	90.61 ± 1.0212	#8116	362.1	91.17 ± 0.5099	#8030	373.1
		0.05	86.15 ± 1.9658	#7903	361.2	91.08 ± 0.6124	#8120	411.6
		0.10	85.33 ± 1.6824	#8852	360.8	89.90 ± 0.8302	#8105	399.6

benefits from the l_1 regularization, we will compare the following strategies for starting points:

A. Randomly set 50 % of the components of the initial solution to be 0.
B. Start from a solution of C-SVM.

Each data set is tested 10 times, and the results about the average test accuracy and its standard deviation, the average number of support vectors and training time are recorded in Tables 1 and 2, which concern the linear kernel and the Gaussian kernel, respectively. Considering the runtime of problems with more than 10,000 samples is quite long, we configure the parameters based on a smaller subset. For the other situations, we tuned the parameters by 10-fold cross-validation.

From the results in Tables 1 and 2, we can see that in most of the cases, Strategy **B** overwhelms **A** in sparsity, though it requires more runtime. Generally speaking, Algorithm 2 with Gaussian kernel performs better than linear kernel in the test accuracy, while it needs more training time especially for bigger data. This is because that for small problems there

Table 4 Comparisons of C-SVM and the proposed coordinate descent algorithm with Gaussian kernel in average test accuracy (Acc.), number of support vectors (SV.) and training time (Time.)

Data	m	r	C-SVM			Alg. 2		
			Acc. (%)	SV.	Time. (s)	Acc. (%)	SV.	Time. (s)
Spect	100	0.00	91.98 ± 2.1034	#67	0.030	91.98 ± 0.0000	#28	0.021
		0.05	90.69 ± 0.9823	#70	0.031	88.88 ± 0.5942	#28	0.014
		0.10	88.30 ± 1.2350	#71	0.031	89.41 ± 0.4957	#32	0.015
Brst.	100	0.00	95.37 ± 0.0408	#47	0.036	94.80 ± 0.0839	#21	0.022
		0.05	93.15 ± 0.3355	#47	0.038	94.23 ± 1.0591	#21	0.021
		0.10	90.08 ± 0.0465	#50	0.038	91.12 ± 0.7639	#18	0.016
Hbrn.	200	0.00	73.96 ± 1.4381	#133	0.063	75.47 ± 0.0000	#111	0.023
		0.05	72.64 ± 1.4057	#118	0.049	76.57 ± 1.9193	#101	0.032
		0.10	72.64 ± 1.7203	#109	0.050	73.45 ± 1.7928	#121	0.032
IJCNN.	1000	0.00	81.89 ± 4.0633	#207	0.149	80.87 ± 0.0001	#556	0.264
		0.05	77.03 ± 3.2900	#370	0.211	78.55 ± 0.6342	#219	0.249
		0.10	73.92 ± 3.0216	#420	0.214	77.02 ± 2.3518	#210	0.273
IJCNN.	4959	0.00	85.53 ± 2.5086	#886	1.758	87.34 ± 2.0605	#909	6.089
		0.05	84.73 ± 2.6980	#1336	2.746	86.19 ± 2.5972	#888	4.155
		0.10	84.60 ± 2.8593	#1008	2.036	85.43 ± 2.6340	#890	3.826
IJCNN.	9989	0.00	86.08 ± 3.9862	#2302	11.52	88.33 ± 1.5710	#1697	14.94
		0.05	86.31 ± 3.1208	#2418	12.03	86.56 ± 2.4908	#1696	18.03
		0.10	71.59 ± 3.1520	#2409	11.98	86.01 ± 1.4071	#1702	12.98
IJCNN.	19,992	0.00	89.23 ± 1.8012	#5986	40.39	91.96 ± 3.1322	#3259	67.18
		0.05	83.15 ± 1.3985	#5921	40.01	86.02 ± 1.4738	#3306	68.42
		0.10	81.08 ± 1.1099	#6033	41.52	84.22 ± 0.3614	#3277	65.91
IJCNN.	49,999	0.00	91.12 ± 0.0382	#7964	370.5	93.10 ± 0.0008	#8122	425.3
		0.05	86.31 ± 0.3421	#8016	365.0	89.26 ± 1.2645	#8193	422.7
		0.10	87.08 ± 0.6890	#8125	371.2	88.24 ± 1.0027	#8108	422.2

is no obvious difference for Algorithm 2 to deal with different kernels, but much time will be spent on computing the kernel when the caching method is used.

4.2 Comparison of Two Algorithms

In order to verify the robustness of ramp-LPSVM solved by Algorithm 2, we randomly flip the signs of the labels of r percent of the training data, where $r = 0, 5, 10\,\%$. Cross-validation is applied in this experiment, and each experiment is repeated 10 times with starting points generated by C-SVM. The average test accuracy, number of support vectors, and training time are recorded in Tables 3 and 4.

From Tables 3 and 4, we can see that Algorithm 2 trains the data faster than C-SVM when the problem scale is quite small (e.g. below 1000), while for larger problems, C-SVM is faster. Even though, there are some other advantages of Algorithm 2 over C-SVM. First, the test accuracy of Algorithm 2 is better than C-SVM in most cases, since the minimization of subproblems implies global search ability to some extent. Second, within a similar test

accuracy level, Algorithm 2 enjoys more sparsity which will contribute to the classification efficiency. In addition, when the number of outliers increases, their influences are less than that in C-SVM, which owes to the robustness induced by ramp loss. Besides, as the randomness exists in the proposed algorithm and also the experiments, we record the standard variance of test accuracy, which shows the stability of Algorithm 2.

5 Conclusion

In this paper, we proposed a modified coordinate descent algorithm for ramp-LPSVM. Since the coordinate descent algorithm updates one single entry of the variable each time, the reduced one-variable piecewise linear subproblems can be solved efficiently by linearizing the concave part. In order to make it converge fast, we considered some implementation tricks to improve the efficiency of the algorithm applied. Experimental performances in robustness, sparsity and fast convergence are confirmed on benchmark data sets.

Acknowledgments This work is supported in part by the National Natural Science Foundation of China, 61134012, 61473165, and the National Basic Research Program of China, 2012CB720505. Xiaolin Huang and Johan Suykens acknowledge support from KU Leuven, the Flemish government, FWO, the Belgian federal science policy office and the European Research Council (CoE EF/05/006, GOA MANET, IUAP DYSCO, FWO G.0377.12, BIL with Tsinghua University, ERC AdG A-DATADRIVE-B).

References

1. An LTH, Tao PD (2005) The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann Oper Res 133(1–4):23–46
2. Aronszajn N (1950) Theory of reproducing kernels. Trans Am Math Soc 68(3):337–404
3. Bache K, Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
4. Bartlett PL, Mendelson S (2003) Rademacher and Gaussian complexities: risk bounds and structural results. J Mach Learn Res 3:463–482
5. Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory., COLT '92ACM, New York, NY, USA, pp 144–152
6. Brooks JP (2011) Support vector machines with the ramp loss and the hard margin loss. Oper Res 59(2):467–479
7. Carrizosa E, Nogales-Gmez A, Morales DR (2014) Heuristic approaches for support vector machines with the ramp loss. Optim Lett 8(3):1125–1135
8. Chang KW, Hsieh CJ, Lin CJ (2008) Coordinate descent method for large-scale L2-loss linear support vector machines. J Mach Learn Res 9:1369–1398
9. Collobert R, Sinz F, Weston J, Bottou L (2006) Trading convexity for scalability. In: Proceedings of the 23rd international conference on Machine learning. ACM, pp 201–208
10. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
11. Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33(1):1
12. Fung GM, Mangasarian OL (2004) A feature selection newton method for support vector machine classification. Comput Optim Appl 28(2):185–202
13. Hildreth C (1957) A quadratic programming procedure. Naval Res Logist Q 4(1):79–85
14. Hsieh CJ, Chang KW, Lin CJ, Keerthi SS, Sundararajan S (2008) A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th International Conference on Machine Learning. ICML '08ACM, New York, NY, USA, pp 408–415
15. Huang X, Shi L, Suykens JAK (2014) Ramp loss linear programming support vector machine. J Mach Learn Res 15:2185–2211
16. Huang X, Xu J, Mu X, Wang S (2012) The hill detouring method for minimizing hinging hyperplanes functions. Comput Oper Res 39(7):1763–1770

17. Lee CP, Lin CJ (2013) A study on L2-loss (squared hinge-loss) multiclass SVM. Neural Comput 25(5):1302–1323
18. Liu Y, Zhang H, Park C, Ahn J (2007) Support vector machines with adaptive L_q penalty. Comput Stat Data Anal 51(12):6380–6394
19. Liu Y, Wu Y (2006) Optimizing ψ-learning via mixed integer programming. Stat Sin 16(2):441
20. Luo Z, Tseng P (1992) On the convergence of the coordinate descent method for convex differentiable minimization. J Optim Theory Appl 72(1):7–35
21. Mangasarian OL (2002) A finite Newton method for classification. Optim Methods Softw 17(5):913–929
22. Mangasarian OL (2006) Exact 1-norm support vector machines via unconstrained convex differentiable minimization. J Mach Learn Res 7:1517–1530
23. Mangasarian OL, Musicant DR (1999) Successive overrelaxation for support vector machines. IEEE Trans Neural Netw 10(5):1032–1037
24. Mazumder R, Friedman JH, Hastie T (2011) Sparsenet: coordinate descent with nonconvex penalties. J Am Stat Assoc 106(495):1125–1138
25. McAllester D, Keshet J (2011) Generalization bounds and consistency for latent structural probit and ramp loss. Adv Neural Inf Process Syst 24:2205–2212
26. Plate JC (1999) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges CJ, Smola AJ (eds) Advances in kernel methods: support vector learning. MIT press, Cambridge, pp 185–208
27. Schmidt M, Fung G, Rosales R (2007) Fast optimization methods for L1 regularization: a comparative study and two new approaches. In: Machine learning: european conference on machine learning 2007, vol. 4701. Springer, Berlin, pp 286–297
28. Shalev-Shwartz S, Tewari A (2011) Stochastic methods for l_1-regularized loss minimization. J Mach Learn Res 12:1865–1892
29. Shao YH, Deng NY (2012) A coordinate descent margin based-twin support vector machine for classification. Neural Netw 25:114–121
30. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
31. Tseng P, Yun S (2009) A coordinate gradient descent method for nonsmooth separable minimization. Math Program 117(1–2):387–423
32. Wang L, Jia H, Li J (2008) Training robust support vector machine with smooth ramp loss in the primal space. Neurocomputing 71(13):3020–3025
33. Wu Y, Liu Y (2007) Robust truncated hinge loss support vector machines. J Am Stat Assoc 102(479):974–983
34. Xu L, Crammer K, Schuurmans D (2006) Robust support vector machine training via convex outlier ablation. In: Proceedings of the 21st national conference on artificial intelligence, AAAI'06. AAAI Press, pp 536–542
35. Yuan GX, Chang KW, Hsieh CJ, Lin CJ (2010) A comparison of optimization methods and software for large-scale L1-regularized linear classification. J Mach Learn Res 11:3183–3234
36. Zhu J, Rosset S, Tibshirani R, Hastie TJ (2004) 1-norm support vector machines. Adv Neural Inf Process Syst 16:49–56