Robust Support Vector Machine Using Least Median Loss Penalty *

Yifei Ma∗ Li Li∗ Xiaolin Huang∗ Shuning Wang∗

∗ Department of Automation, Tsinghua University
Tsinghua National Laboratory for Information Science and Technology
Beijing, 100084, China
(e-mails: magifei1012@gmail.com, li-li@mail.tsinghua.edu.cn,
huangxl06@mails.tsinghua.edu.cn, swang@mail.tsinghua.edu.cn)

Abstract: It is found that data points used for training may contain outliers that can generate unpredictable disturbance for some Support Vector Machines (SVMs) classification problems. No theoretical limit for such bad influence is held in traditional convex SVM methods. We present a novel robust misclassification penalty function for SVM which is inspired by the concept of “Least Median Regression”. In our approach, total loss penalty in training is measured by the summation of two median hinge losses, each for a different class. We also propose a “Rank and Convex Procedure” to optimize our tasks. Though our approach is heuristic, it is faster than other known robust methods, such as SVM with Ramp Loss Penalty.

Keywords: Statistical data analysis, Learning theory, Pattern recognition, Robust identification, Support vector machine.

1. INTRODUCTION

1.1 Robust SVM

The Support Vector Machine (SVM) is a binary classification tool developed by Cortes and Vapnik (1995). It is characterized by its fast speed, good generalization ability, and unique solutions when convexity retains (Steinwart and Christmann, 2008). However, in all of its recent developments we surveyed (section 1.2), none has a robustness guarantee (which describes how the classifier works in the presence of outliers) under the criteria of breakdown point, which has been widely used in robust regression (section 1.3). Moreover, our algorithm is shown to run faster than most other robust classification methods, especially the L2-regularized Ramp Loss SVM (Collobert et al., 2006).

On the one hand, outliers in the training set prevent us from learning the true classification rules in practice. For example when SVM is used for intrusion detection in computer systems, Hu (2003) suggested that in the assumed normal period used for training, intrusions may already occur on the way. On the other hand, SVM is used for Outlier Detection through single class classification. Chandola et al. (2007) did a survey of Outlier Detection.

To better illustrate the need for robust SVM, we may see the toy example in Figure 1.

1.2 Known Robust SVM Methods

Some of the methods below are not designed for robustness, but can be used as a robust approach. Other approaches achieve robustness by modifying the loss penalty term in SVM objective function.

Wang et al. (2008) and Wu and Liu (2011) proposed a fast probability estimation method using SVM, where decision rules for different posterior probabilities can be simulated by SVM results when multipliers for loss penalties are set differently on class +1 margin violation and class −1 margin violation. However, the focus in these methods is on probability estimation rather than robustness since they identify outliers after the estimation is done and their speed is slower than traditional SVMs.

An earlier attempt (Song et al., 2002) allowed a small weight (and consequently a big margin) in areas far from the geologic centers of the training data points. However, model assumptions for data distribution are needed.

The Ramp Loss-SVM or ψ-Learning achieves robustness by setting a limit on maximal penalty for a single point violating its supposed margin. This can be achieved by...
Subtraction of two parallel Hinge Loss functions or by creating some piecewise linear functions (Collobert et al., 2006; Ertekin et al., 2010; Liu et al., 2005). The major disadvantage found in these methods is the lack of convexity, i.e. no known method can guarantee a global minimum in polynomial time. Xu et al. (2006) performed mathematical treatments to regain a semi-definite property under some relaxation, whose reformed problem was not yet scalable. Implementations of Ramp Loss-SVM or ψ-Learning on a Concave-Conex Procedure (CCCP) can be found in most of the above papers.

1.3 Breakdown Point & Least Median Squares Regression

Least Median Squares (LMS) is a robust criterion proposed by Rousseeuw (1984) which is used in regression field to resolve a similar robust problem. LMS finds a regression line with least median squared error of sampling data. LMS is proved to be the most robust criterion, when measured by the concept of breakdown point (Donoho and Huber, 1982). The reason can be explained that a few outliers with large errors cannot affect the value of median squared error. Inspired by the idea of LMS, a novel robust SVM method is proposed in this paper.

When LMS is used for linear regression, the regression problem can be solved by some exact algorithms (Steele and Steiger, 1986; Stromberg, 1993; Agull, 1997). Unfortunately, when the data set grows up to some extent, the computing time of these algorithms will expose. Therefore, Rousseeuw and Hubert (1997), Olson (1997), and Chakraborty and Chaudhuri (2008) proposed some approximation algorithms with less computation load. However, all these algorithms can only solve linear regression problems with LMS criterion. Therefore, when applying LMS in SVM, we will also introduce a new algorithm.

1.4 Class Conditional Median Loss-SVM Approach

The main contribution in our work is a Class Conditional Median Loss function (CCML) for the SVM loss penalty measurement. The total loss penalty for all data points is measured by the summation of the median\(^1\) individual hinge losses in each of the two classes. Data points containing a bigger loss are likely to serve as outliers. Moreover, the classification hyperplane remains sparse, yielding a fast training and testing speed comparable to (or even faster than) CCCP, the known fastest robust method.

While discussion and experiments are included in this paper, our main focus is in the mathematical and pseudo-code descriptions of the algorithms and their efficacy. Let us see a benchmark problem comparing Hinge Loss-SVM, Ramp Loss-SVM and CCML-SVM in Fig. 2.

2. FORMULATION OF SVM OPTIMIZATION WITH MEDIAN LOSS PENALTY

2.1 General Formulation of SVM

Suppose the data is expressed as \((x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}\), \(\forall i \in \Omega = \{1, ..., n\}\), where \(d\) is the finite total number of features and \(y_i\) is the class label of \(x_i\). By the above expression, the training data contains only two classes, denoted by their \(y\) attribute as class +1 and class −1. All subscripts of data points in class +1 are denoted by \(\Omega_+ \triangleq \{i \in \Omega, y_i = +1\}\) and in class −1 by \(\Omega_-\). Similarly, \(\Omega_{\text{sgn}}\) stands for either one of them provided \(\text{sgn} = +\) or −.

Based on any continuous function \(f : \mathbb{R}^d \rightarrow \mathbb{R}\), a binary decision can be made that classifies any data point \(x\) to class +1 if \(f(x) > 0\) or class −1 otherwise. This function \(f(x)\) is called a decision function. Given a decision function \(f(x)\) and any data point \((x_0, y_0)\), we measure the level of misclassification (called loss) by \(L(y_0 f(x_0))\), where \(L : \mathbb{R} \rightarrow [0, +\infty)\) is called a Loss Function. A pair of decision margins are defined by the equation \(f(x) = \pm 1\).

SVM learns a decision rule from an optimization which balances margin enlargement and train set loss supression. In linear case, decision function \(f(x) = \omega x + b\), where \((\omega, b)\) are parameters to be learnt. Since \(2/\|\omega\|\) shows the width of the margin, \(\frac{1}{2} \|\omega\|^2 + C \sum_{i \in \Omega} L(y_i (\omega x_i + b))\) is to be minimized. Here \(C\) is the balancing parameter.

\(^1\) or of some other quantile
In general some kernel function \( k: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \) is used to measure the distance of any two data points in \( \mathbb{R}^d \), which results in the construction of a so-called reproducing kernel Hilbert Space (RKHS) \( \mathcal{H} \triangleq \{ k(\cdot, \cdot) : x \in \mathbb{R}^d \} \). Decision function \( f(x) \) is restricted to the form \( f(x) = f_{H}(x) + b \), where \( f_{H}(\cdot) \in \mathcal{H} \) and \( b \) is a real constant. SVM here minimizes \( \frac{1}{2}\| f_{H} \|_{\mathcal{H}}^2 + C \sum_{i \in \Omega} L(y_{i}, f(x_{i})) \).

### 2.2 Traditional Hinge Loss-SVM

In the traditional Hinge Loss-SVM, the loss function for each data point is \( L_{\text{hinge}}(y_{i}, f(x_{i})) = \max(0, 1 - y_{i} f(x_{i})) = [1 - y_{i} f(x_{i})]^+ \). The loss term of SVM goal function and the SVM optimization problem are respectively,

\[
\begin{align*}
\text{Total} & \quad L_{\text{hinge}} = C \sum_{i \in \Omega} [1 - y_{i} f(x_{i})]^+ \quad (1) \\
\min_{f} & \quad \frac{1}{2}\| f \|^{2} + C \sum_{i \in \Omega} [1 - y_{i} f(x_{i})]^+ \quad (2)
\end{align*}
\]

### 2.3 Class Conditional Median Loss-SVM (CCML-SVM)

**Class Conditional Median Loss Penalty** We change the traditional Hinge Loss penalty term (1) to a novel Class Conditional Median Loss penalty term (CCML), \(^2\)

\[
\text{CCML} = C \cdot \frac{1}{2} \text{median}[1 - (1)f(x_{i})]^+ + C \cdot \frac{1}{2} \text{median}[1 - (-1)f(x_{i})]^+
\]

where \( \text{median} \) in general stands for a function that maps a list of values \( \{\phi[i], i = 1, ..., N\} \) to its \( \lfloor \frac{N}{2} \rfloor + 1 \)-th maximal value \( \phi^* \). \(^3\)

In (3), it is used to find in \( \{[1 - y_{i} f(x_{i})]^+, i \in \Omega_{\text{sgn}}\} \) its \( \lfloor \frac{|\Omega_{\text{sgn}}|}{2} \rfloor + 1 \)-th maximum, where \( \text{sgn} = + \) stands for class +1 and \( \text{sgn} = - \) for class -1.

**Median** \(^4\) in this paper may also be replaced by some other upper quantiles in order to implement noise level control. A quantile is denoted here as the rank in percentage of an extracted single value in the given list of values it belongs to. For example, a 25\% quantile means the \( \lfloor 0.25 \times |\Omega_{\text{sgn}}| \rfloor + 1 \)-th maximum from the given list \( \{[1 - y_{i} f(x_{i})]^+, i \in \Omega_{\text{sgn}}\} \). It may not exceed 50\%.

**Balancing Constraint** The SVM solution function \( f(x) = \omega x + b \) or \( f(x) = \sum_{i} \alpha_{i} k(x, x_{i}) + b \) has a constant term \( b \), which can be adjusted to balance the loss penalty for both classes. Hence we may add an additional constraint to (3),

\[
\text{median}[1 - (1)f(x_{i})]^+ - \text{median}[1 - (-1)f(x_{i})]^+ = 0. \quad (4)
\]

\(^2\) Class Conditional is used to remind the reader that our Median Loss is Class based. CCML is not a Loss Function but rather the term to replace the loss penalty term in the SVM goal function.

\(^3\) There is a difference between our definition of median and the statistical definition when \( N \) is even, but this difference is trivial.

\(^4\) For simpler notation, we still use the term median when we actually mean another specified quantile in this paper.

**Complete Formulation** Taking (3) as loss penalty and (4) as constraint, our CCML-SVM optimization goal is,

\[
\begin{align*}
\min_{f(x)} & \quad \frac{1}{2}\| f \|^{2} + C \cdot \frac{1}{2} \text{median}[1 - (1)f(x_{i})]^+ \\
& \quad + C \cdot \frac{1}{2} \text{median}[1 - (-1)f(x_{i})]^+ \\
\text{s.t.} & \quad \text{median}[1 - (1)f(x_{i})]^+ = \text{median}[1 - (-1)f(x_{i})]^+.
\end{align*}
\]

### 3. SOLUTION OF CCML-SVM

#### 3.1 Mixed Integer-Convex Description

In order to form a programmable problem, we need to transform (5) into a mixed integer-convex optimization task presented in Theorem 1. The following two Lemmas lead to our result there.

**Lemma 1.** Problem (5) is equivalent to the problem below,

\[
\begin{align*}
\min_{f,R_+,R_-} & \quad \frac{1}{2}\| f \|^{2} + C \cdot \max_{|R_{\Omega_{+}}| \leq |R_{\Omega_{-}}|} \text{median}[1 - y_{i} f(x_{i})]^+ \\
\text{s.t.} & \quad 1 - y_{i} f(x_{i}) \leq 1 - y_{i} f(x_{j}), \forall i \in R_{+} \cup R_{-}, \forall j \in H_{+} \cup H_{-},
\end{align*}
\]

where \( R_{+} \) and \( H_{+} \) donate an arbitrary partition of \( \Omega_{+} \) (i.e., \( \Omega_{+} = R_{+} \cup H_{+} \) and \( R_{+} \cap H_{+} = \emptyset \)) that satisfies the constraints. \( R_{-} \) and \( H_{-} \) are defined similarly.

**Proof.** One way to evaluate a median function \( \text{median}[\phi[i], i = 1, ..., N] \) is to partition the universe set \( A = \{1, ..., N\} \) into two subsets \( A_{R} = \{\sigma_{1}, ..., \sigma_{k}\} \) and \( A_{H} = \{\sigma_{k+1}, ..., \sigma_{N}\} \), where \( k = \lfloor \frac{N}{2} \rfloor \) and \( \{\sigma_{1}, ..., \sigma_{N}\} \) is a permutation of \( \{1, ..., N\} \).

If \( \phi[\sigma_{i}] \leq \phi[\sigma_{j}] \) for all pairs \( \{\sigma_{i} \in A_{R}, \sigma_{j} \in A_{H}\} \) (this condition exists for example when \( \sigma_{i} \) is chosen such that \( \phi[\sigma_{i}] \)’s fall into an ascending order), it can be concluded that the maximal \( \phi[\sigma_{j}] \) in \( A_{R} \) is ranked in value after \( |A_{R}| \) elements but before the rest \((|A_{R}|-1)\) elements. It is then \( (|A_{H}|+1)-\text{th} \) (the \( \lfloor \frac{N}{2} \rfloor + 1 \)-th) maximum.

Applying the above general treatment of median to the specific CCML penalty, we get the following,

\[
\begin{align*}
\text{median}[1 - y_{i} f(x_{i})]^+ = \max_{|R_{\Omega_{+}}| \leq |R_{\Omega_{-}}|} [1 - y_{i} f(x_{i})]^+ \\
\text{s.t.} & \quad 1 - y_{i} f(x_{i}) \leq 1 - y_{i} f(x_{j}), \forall i \in R_{+}, \forall j \in H_{+}
\end{align*}
\]

where \( R_{+} \) and \( H_{+} \) denote an arbitrary partition of \( \Omega_{+} \) satisfying the constraints. \(^5\) Via replacing subscripts “+” by “-”, a similar result may be derived for class -1. We denote this counterpart by (7).

\(^5\) If \( \phi^* \) is used to denote the optimal solution of the MIP problem (7), then the data points with penalty values (by evaluating \( 1 - y_{i} f(x_{i})^+ \) less than \( \phi^* \) are contained in \( R_{+} \) and those greater than \( \phi^* \) are contained in \( H_{+} \). There might be multiple data points whose penalty values are exactly at \( \phi^* \). In this case, those points are randomly picked up by \( R_{+} \) and \( H_{+} \) and have no influence on \( \phi^* \).
Constraint (4) makes the objective in (7) and (7’) equal. Hence they are combined to form a single MIP (6).

Lemma 2. The optimization task (6) has the same minimized goal value as the following task,

\[
\min_{f, R_+, R_-} \left( \frac{1}{2} \| \omega \|^2 + C \max_{i \in R_+ \cup R_-} \left[ 1 - y_i f(x_i) \right]^{+} \right) \\
\text{s.t.} \quad |R_+| = [50\% \times |\Omega_+|], \quad |R_-| = [50\% \times |\Omega_-|]
\]

where \( R_+ \) denotes a subset of \( \Omega_+ \) and \( R_- \) a subset of \( \Omega_- \).

Proof. The only difference between (6) and (8) is that (6) has an additional constraint,

\[
1 - y_i f(x_i) \leq 1 - y_j f(x_j), \quad \forall i \in R_+ \cup R_- \text{ and } \forall j \in H_+ \cup H_-. \tag{9}
\]

Clearly, relaxed from the constraint (9), minimized goal value in (8) ≤ minimized goal target value in (6).

In order to prove that the minimized goal value in (6) is no greater than (8), we only need to construct a solution feasible in (6) (satisfying (9)) from an optimum solution in (8) and the goal value does not increase. Given that, minimized goal in (8) ≥ goal value with our constructed solution ≥ minimized goal in (6).

Followed is the construction.

Suppose one optimum in (8) is found at \((f, R_+, R_-) = (f^*, R^*_+, R^*_-)\). Denote \( i^*_+ \) as a subpart of the maximal penalty in \( R^*_+ \), i.e., \( [1 - y_{i^*_+} f^*(x_{i^*_+})]^{+} = \max_{i \in R^*_+} [1 - y_i f(x_i)]^{+} \). Also denote \( i^*_- \) the counterpart in \( R^*_- \).

Assume \([1 - y_{i^*_+} f^*(x_{i^*_+})]^{+} = [1 - y_{i^*_-} f^*(x_{i^*_-})]^{+} \). Otherwise, by changing \( b \) in \( f^* \), (8) may be further minimized.

When (9) does not hold as (8) reaches its minimum, there exists at least a pair of \( i \in R^*_+ \cup R^*_- \) and \( j \in H^*_+ \cup H^*_- \) such that \([1 - y_i f(x_i)]^{+} > [1 - y_j f(x_j)]^{+} \). This pair is called a twisted pair because they violate the order between \( R_+ \cup R_- \) and \( H_+ \cup H_- \) described by (9).

The following holds for either \( i \in R_+ \) or \( i \in R_- \) and for either \( j \in H_+ \) or \( j \in H_- \),

\[
[1 - y_{i^*_+} f^*(x_{i^*_+})]^{+} = [1 - y_{i^*_-} f^*(x_{i^*_-})]^{+} \\
\geq [1 - y_i f(x_i)]^{+} > [1 - y_j f(x_j)]^{+}.
\]

In the case \( j \in H^*_+ \), we change \( R^*_+ \) to \( \tilde{R}^*_+ \triangleq (R^*_+ \setminus \{ i^*_+ \}) \cup \{ j \} \) and \( H^*_+ \) to \( \tilde{H}^*_+ \triangleq (H^*_+ \setminus \{ j \}) \cup \{ i^*_+ \} \).

Now \( \max_{i \in R^*_+ \cup R_-} [1 - y_i f(x_i)]^{+} = [1 - y_{i^*_+} f^*(x_{i^*_+})]^{+} \geq \max_{i \in \tilde{R}^*_+ \cup R_-} [1 - y_i f(x_i)]^{+} \), which means that the goal value in (8) does not increase. It cannot decrease either since it is already the minimum. The new pair \((\tilde{R}^*_+, \tilde{R}^*_-)\) is in fact also an optimum solution for (8).

For the case \( j \in H^*_- \), we swap \( i^*_+ \in R^*_+ \) with this \( j \) and apply a similar analysis like the above.

After the construction of a new \( \tilde{R}^*_+ \) or \( \tilde{R}^*_- \), the number of twisted pairs \( |(i, j) : [1 - y_i f(x_i)]^{+} > [1 - y_j f(x_j)]^{+}, i \in \tilde{R}^*_+ \cup \tilde{R}^*_-, j \in H_+ \cup H_- \) \) decrease. Hence, after finite steps repeating the above construction (from “When (9) does not hold ...”), twisted pairs will disappear.

Concluding Lemma 1 and 2, our coding-friendly MIP tasks are as followed.

Theorem 1. Optimization task (5) is equivalent to the Mixed-Integer Programming task (8).

Corollary 1. Optimization task (5) is equivalent to the following MIP task described with a slack variable \( \xi \),

\[
\min_{f, R_+, R_-, \xi} \left( \frac{1}{2} \| \omega \|^2 + C \xi \right) \\
\text{s.t.} \quad \left\{ \begin{array}{l}
1 - y_i f(x_i) \leq \xi, \quad \forall i \in R_+ \cup R_- \\
\xi \geq 0
\end{array} \right.
\]

where \( R_+ \) denotes a subset of \( \Omega_+ \) and \( R_- \) a subset of \( \Omega_- \).

3.2 Dual Problem

When \( R_+ \) and \( R_- \) are fixed in (10), MIP becomes a quadratic optimization problem. We call the problem for fixed \( R_+ \approx R_+ \cup R_- \) the convex part in (10).

The Lagrangian for this convex part is,

\[
L_P = \frac{1}{2} \| \omega \|^2 + C \xi - \sum_{i \in R} \alpha_i (1 - y_i f(x_i) + \beta - \xi),
\]

where \( \alpha_i \geq 0 \) and \( \beta \geq 0 \) are dual variables.

For the above convex part, the Karush-Kuhn-Tucker (KKT) condition is,

\[
\begin{align*}
\frac{\partial L_P}{\partial \omega} &= \omega - \sum_{i \in R} \alpha_i y_i x_i = 0, \\
\frac{\partial L_P}{\partial \beta} &= -\sum_{i \in R} \alpha_i y_i = 0, \\
\frac{\partial L_P}{\partial \xi} &= -\sum_{i \in R} \alpha_i + C(1 - \beta) = 0, \\
\alpha_i (1 - y_i (\omega x_i + b) - \xi) &= 0, \\
\beta &= 0, \\
\alpha_i &\geq 0,
\end{align*}
\]

(i \( R \) is omitted for all conditions and summations).

The dual problem for convex part of (10) is then,

\[
\max_{\alpha_i} \left( \sum_{i \in R} \alpha_i - \sum_{i,j \in R} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right) \\
\text{s.t.} \quad \left\{ \begin{array}{l}
\alpha_i \geq 0, \quad \forall i \in R, \\
\sum_{i \in R} \alpha_i \leq C, \\
\alpha_i y_i = 0
\end{array} \right.
\]

where \( k(x_i, x_j) \) stands for a specific kernel function measuring the distance between \( x_i \) and \( x_j \).

A relationship between dual convex part and primal convex part can be derived from KKT criteria (11) and the fact that the optimal goal values of (10) and (12) are equal.

3.3 Rank and Convex Procedure (RCP)

A reasonable solution of (10) may be achieved by Algorithm 1.

---

6 for example an Euclidean norm distance for linear decision or a Gaussian RBF for a typical nonlinear decision.
Algorithm 1: Rank and Convex Procedure (RCP)

input: \( \Omega^0 \) denoting a subset of training set \( \Omega \) that contains 20% random data from each class or \( f^0(x) \) denoting an initial decision function.

output: A convergent decision function \( f(x) \).

initialize

Train 20% random data from each class with Hinge Loss-SVM for \( f^0(x) = \sum_{j \in \Omega^0} \alpha_{ij} y_j k(\mathbf{x}_i, \mathbf{x}_j) + b^0 \).

repeat

\[ R = \text{new } R_{\text{sgn}} \text{ by evaluating } f^{t-1}(\mathbf{x}_i), i \in \Omega_{\text{sgn}} \text{ and sorting them (} \text{sgn} = \{+\text{or} -\}. \]

Solve optimum \( \alpha^t \) in dual convex part

\[ \max_{\alpha_{ij}} \left( \sum_{i \in R} \alpha_i - \frac{1}{2} \sum_{i,j \in R} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \right) \]

s.t.

\[ \sum_{i \in R} \alpha_i \leq C, i \in R \]

Back to primal convex part

\[ \ell^t = \frac{1}{c} \left( \sum_{i \in R} \alpha_i - \sum_{i,j \in R} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \right) \]

\[ b^t = 1 - \ell^t - y_i \sum_{j \in R} \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j), \forall \alpha_i > 0, i \in R \]

\[ f^t(x) = \sum_{i \in R} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b^t \]

until \( R^{t-1} = R^t \) or \( t \) exceeds the iteration limit.

Theorem 2. RCP converges in finite steps.

Proof. For the goal function in (8), \( J(f, R_+, R_-) = \frac{1}{2} ||\omega||^2 + C \max_{x \in R} |1 - y_i f(x_i)|^2 \). “C step” decreases its value directly by optimizing \( f(\cdot) \) and “R step” does not increase its value. Also, for each choice of “R”, the C step is a convex problem and has a unique optimum. Since the combination of \( R \) is finite (because \( R \subseteq \Omega \)), RCP will converge in finite steps.

3.4 RCP+ Take Advantage of Traditional Solvers

Algorithm 1 solves our goal (5) through (8) directly. However, from observations of RCP, in some situations we can improve its coding feasibility in RCP+ form.

Since we have neglected half of (or some other percentage of) data points from each class for consideration in C step in RCP, it appears that the remained training data points (contained in \( R_+ \) or \( R_- \)) are completely separable. Hence, a hard margin may be applied to the remaining \( R_+ \cup R_- \) subset. This can be implemented with \( C \to \infty \) for any kind of loss function. Alternative C step adapted from total hinge loss\(^7\) in \( R_+ \cup R_- \) is listed in Table 1.

3.5 Heuristic Methods for a Better Solution

There are two mechanisms for heuristic methods to fiddle with the input of RCP. In one way, we can heuristically combine or modify convergent decision functions \( f(\cdot) \) from the last RCP output and begin a new RCP with the revised decision functions. In another way, we may choose different initializing subsets \( \Omega^0 \) heuristically according to the last RCP outputs and begin a new RCP with the new subset. In the latter mechanism, we may set bigger chances for the correctly classified points such that the convergent decision function in the following iteration will not change greatly. In general, the former mechanism is better in exploration while the latter in exploitation.

4. PROPERTIES

4.1 Choice of Class Conditional Quantile and Robustness

The choice of conditional quantiles \( |R_{\text{sgn}}|/|\Omega_{\text{sgn}}| \) is a trade-off between robustness and accuracy. This is shown in Figure 3. A bigger quantile means better accuracy but also smaller margin and less tolerance on outliers, putting SVM at risk of overfitting. We suggest that a proper quantile allow twice the neglecting percentage as the error rate of a typical classifier.

\[ f^t(x) = \sum_{i \in R} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b^t \]

Table 1. Alternative C+ Step for RCP+

<table>
<thead>
<tr>
<th>Name</th>
<th>Alternative Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C+ Step:</td>
<td>Solve optimum ( \alpha^t ) in dual convex part</td>
</tr>
<tr>
<td>Hard Margin</td>
<td>[ \max_{\alpha_{ij}} \left( \sum_{i \in R} \alpha_i - \frac{1}{2} \sum_{i,j \in R} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \right) ]</td>
</tr>
<tr>
<td>for RCP+</td>
<td>[ R^t = R^t_+ \cup R^t_- ]</td>
</tr>
</tbody>
</table>

\( R \rightarrow \infty \) or \( t \) exceeds the iteration limit.

---

\( \text{Fig. 3. Decision line for toy 1-D data under different quantiles. 250 points are generated randomly with } |\Omega_+| \approx 3|\Omega_-|. \) (a) shows class conditional distribution \( p_{\mathbf{X}}(\mathbf{x}|1) \) and \( p_{\mathbf{X}}(\mathbf{x}|-1) \) by red and blue curves respectively. (b) shows the posterior possibility of \( P(Y = 1|\mathbf{X} = \mathbf{x}) \) (red) and \( P(Y = -1|\mathbf{X} = \mathbf{x}) \) (blue). (c) is a Hinge Loss-SVM result simulating Bayesian decision function \( f_{\text{dp}}(\mathbf{x}) = P(Y = 1|\mathbf{X} = \mathbf{x}) - P(Y = -1|\mathbf{X} = \mathbf{x}) \). (d), (e), (f) are CCML-SVM results for different \( |R_{\text{sgn}}|/|\Omega_{\text{sgn}}| \) values (90%, 70%, and 50% respectively). Only horizontal position of each data point is considered as a feature.

\(^7\) Of course, in terms of hard margin, a variety of loss penalty term for SVM may be used.
4.2 Comparison with Ramp Loss for Speed

The Ramp Loss penalty function, or a ConCave-Convex Procedure (CCCP) is shown in Collobert et al. (2006) to be faster than the classical convex approach of Hinge Loss penalty. Algorithm 2 shows their approach.

Algorithm 2: ConCave-Convex Procedure as depicted in Collobert et al. 2008 (for purpose of comparison)

initialize
- Train 20% random data with Hinge Loss-SVM for $f_t^0(x) = w^0 x + b^0$.

repeat

CC Step
- Compute $\beta^t_i = \begin{cases} C, & \text{if } y_i f_{t-1}(x_i) < s \\ 0, & \text{otherwise} \end{cases}$

C Step
- Solve $\alpha^t_i$ for
  \[
  \max_{\alpha^t_i} \left( \sum_{i \in \Omega} \alpha_i - \frac{1}{2} \sum_{i,j \in \Omega} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right) \\
  \text{s.t.} \left\{ \begin{array}{l}
  \sum_{i \in \Omega} \alpha_i y_i = 0 \\
  -\beta^t_i \leq \alpha_i \leq C - \beta^t_i, \quad i \in \Omega \\
  \text{Compute } b^t \text{ using}
  \\
  y_i (\sum_{j \in \Omega} \alpha^t_j y_j k(x_i, x_j) + b^t) = 1, \forall 0 < \alpha_i < C \\
  \text{C Step result is } f^t(x) = \sum_{j \in \Omega} \alpha^t_j y_j k(x, x_j) + b^t
  \end{array} \right. \\
\]

until $\beta^{t-1} = = \beta^t$ or $t$ exceeds iteration limit.

It is clear that CCCP and RCP (or RCP+) follow similar routines, 1) a $C$ Step taking advantage of the SVM convex optimizer, and 2) a $CC$ Step or $R$ Step in deciding which data points to train in the $C$ Step.

Their difference is primarily whether the data sets fed into $C$ Step contain noises. In CCCP, they allow noises to the extent that $s < y_i f(x_i) \leq 1$. In our approach however, noises are preconditioned. Therefore, on similar conditions, the number of Support Vectors are decreased substantially in our approach and speed in $C$ Step increased.

Moreover, if the $C$ Step is done with a Gaussian RBF and the primal result cannot be simplified, the number of SVs reflects the number of implicit $C_i \exp(-\sigma \|x_i\|^2)$ terms and our approach greatly simplifies the result. This fact contributes to a much faster evaluation of $y_i f(x_i)$ in $R$ Step than in $CC$ Step.

5. CONCLUSION

In this paper, we proposed a novel approach, Class Conditional Median Loss-SVM for robust classification. We described its equivalent optimization problems and a RCP+ algorithm to solve them. We also demonstrated its unique properties like adaptivity to different noise levels and even faster speed than ConCave-Convex Procedures. A demo in 2-D benchmark dataset is reported as well.

ACKNOWLEDGEMENTS

We highly appreciate the helpful comments from Dr. Huachun Zhang (Institute of Electronics, Chinese Academy of Sciences), Dr. Jun Xu and Mr. Qiang Meng (Department of Automation, Tsinghua University).

REFERENCES


