Scientific Research Publishing

Applied Mathematics, 2025, 16(5), 429-440
https://www.scirp.org/journal/am
ISSN Online: 2152-7393
ISSN Print: 2152-7385

The SVM Classifier with Quartic Truncated Pinball Loss

Lumiao Wang*, Ziye Liu

Department of Mathematics, Jinan University, Guangzhou, China
Email: *lmiaowang@163.com

How to cite this paper: Wang, L.M. and Liu, Z.Y. (2025) The SVM Classifier with Quartic Truncated Pinball Loss. *Applied Mathematics*, **16**, 429-440.
https://doi.org/10.4236/am.2025.165023

Received: April 7, 2025
Accepted: May 20, 2025
Published: May 23, 2025

Abstract

In the field of machine learning, support vector machine (SVM) is popular for its powerful performance in classification tasks. However, this method could be adversely affected by data containing noise. One of the main reasons is that many loss functions are too sensitive to sample points far from their classes. In this paper, based on the idea of HTPSVM proposed in (Zhu, Song, *et al.*), we introduce the SVM classifier with quartic truncated pinball loss (QTPSVM), which can be solved by Bregman modified second accelerated proximal gradient method (BAPGs). Numerical experiments show that QTPSVM is more effective on many real data.

Keywords

Support Vector Machine, the SVM Classifier with Huberized Truncated Pinball Loss (HTPSVM), the SVM Classifier with Quartic Truncated Pinball Loss (QTPSVM), the Bregman Modified Second APG Method (BAPGs)

1. Introduction

Support vector machine (SVM) is a supervised learning method mainly used for classification problems [1]. It distinguishes data points of two different classes by finding an optimal hyperplane that maximizes the boundary between the two classes. SVM plays an important role in the field of machine learning for its excellent performance and generalization ability. Its applications span multiple subjects, including key areas such as text classification, image recognition, bioinformatics, and financial risk assessment.

The performance of SVM largely depends on the choice of its loss function. Different loss functions are suitable for different data distribution and noise conditions. The SVM with classic hinge loss (HSVM) [2] increases the distance between the nearest points of different categories, but the model is easily affected by noise. To overcome this limitation, the SVM with pinball losses (PSVM) [3] was

proposed, which deducts points from correctly classified points and is insensitive to noise during resampling. However, it sacrifices sparsity and lacks flexibility in adjusting parameters that affect the loss value. In order to overcome these shortcomings, the SVM with generalized pinball loss (GPSVM) [4] was introduced. GPSVM reduces noise sensitivity while retaining sparsity and flexibility of parameter adjustment in the derived solution.

The above mentioned loss functions are convex but not differentiable, so the corresponding unconstrained minimization problems are not differentiable. Therefore, although some first-order and second-order optimization algorithms [5] are theoretically effective and have solid theoretical support, they cannot be directly used to solve such problems. In order to solve the limitation of traditional hinge loss function, researchers introduced smooth hinge loss function (SHSVM) [6] to improve the performance of the model. Similarly, to address the shortcomings of the pinball loss function, researchers proposed a smooth pinball loss function (SPSVM) [7]. Based on the above ideas, the smooth generalized pinball loss function (SGPSVM) [8] was given. These smooth loss functions are differentiable, so that the corresponding models can be solved directly by applying first order and second-order optimization techniques.

Although some progress has been made in the design of loss functions, the above mentioned loss functions are all unbounded, which makes the models face significant challenges when dealing with outlier data points. Outliers are those extreme values that differ significantly from the rest of the data set, and they can be caused by sampling errors or labeling errors. In classification tasks, it is critical to properly handle these outliers, as they can seriously affect the performance of the model. Many SVM models are particularly sensitive to these outliers because their loss functions may grow indefinitely on these points. This phenomenon has led to the development of bounded loss functions, such as the truncated hinge loss function in RSVM [9]. By truncating the loss function, the high values are reset to some constant(s), so that the impact of outliers is under control and the model performance is enhanced.

Recently, Zhu *et al.* [10] has proposed bounded huberied truncated pinball loss function which can reduce the effect of noise in training samples. Since quartic functions are more complex than quadratic functions, they may improve classifier performance, especially when data follows a nonlinear distribution. Based on this idea, we change the Huberized truncated pinball loss function by replacing the quadratic term with the quartic term. The newly proposed quartic truncated pinball function is bounded and differentiable. The Bregman modified second APG method by Wang *et al.* [11] performs well on such problems, so we use BAPGs to solve the support vector machine problem with the quartic truncated pinball function (QTPSVM).

The structure of this paper is as follows: In Section 2, some classical SVM models are reviewed. In Section 3, we introduce the QTPSVM model. Then we discuss the application of the BAPGs algorithm on QTPSVM in Section 4. Numerical experiments on real data are done to verify the effectiveness of QTPSVM in Section

5. The last Section is the summary of this paper.

2. Review on Some Support Vector Machine Models

In the SVM model for binary classification problem, we denote the training dataset as $\{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^n \times \{1,-1\}, i=1,2,3,\cdots,m\}$, where $\boldsymbol{x}_i$ is a n-dimensional predictor variable, $\boldsymbol{y}_i$ is its label and m is the number of samples. The key to solving this problem is to identify a separating hyperplane represented by $w^{\mathrm{T}}\boldsymbol{x}+b=0$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$, with the objective of maximizing the margin and the training error. The SVM classification can be expressed as an unconstrained optimization problem:

$$\min_{w,b} \frac{1}{m}\sum_{i=1}^{m} \ell\left(\boldsymbol{y}_i\left(w^{\mathrm{T}}\boldsymbol{x}_i+b\right)\right) + \mathrm{Penalty}_{\lambda}(w,b),$$

where ℓ denotes the loss function and $\mathrm{Penalty}_{\lambda}(w,b)$ is the penalty (regularization) term, $\lambda>0$ is the regularization parameter, used to balance the loss and the penalty. The decision function $w^{\mathrm{T}}\boldsymbol{x}+b$ is used to determine the class of $\boldsymbol{x}$: $\boldsymbol{x}$ is classified as positive if the the decision function is positive, and negative otherwise.

There are several kinds of penalty terms commonly used in support vector machine problems:

1) The L_2 norm penalty $\frac{\lambda}{2}\|w\|_2^2$. This is a traditional penalty term for various machine learning problems, but it does not perform well on sparse solutions.

2) The L_1 norm penalty $\lambda\|w\|_1$. It has a good effect on sparsity, but it is limited by the number of samples and may over-prune in some highly correlated features [12].

3) The Elastic Net penalty combines the L_1 norm and the L_2 norm [13], integrates the advantages of both terms and overcoming their drawbacks. The commonly used forms are:

$$\lambda_1\|w_1\|_1 + \frac{\lambda_2}{2}\|w_2\|_2^2 \tag{2.1}$$

and

$$\lambda_1\|w_1\|_1 + \frac{\lambda_2}{2}\|w_2\|_2^2 + \frac{\lambda_3}{2}b^2. \tag{2.2}$$

In SVM, the most popular and widely used loss function is the hinge loss function (Rectified Linear Unit function), which is denoted by $\ell_{\mathrm{hinge}}: \mathbb{R} \to \mathbb{R}$, and is defined as follows:

$$\ell_{\mathrm{hinge}}(u) = \max\{0, 1-u\} = \begin{cases} 1-u, & u \le 1, \\ 0, & u > 1. \end{cases}$$

Another commonly used one is the pinball loss function $\ell_{\mathrm{pinball}}: \mathbb{R} \to \mathbb{R}$,

$$\ell_{\mathrm{pinball}} = \begin{cases} 1-u, & u \le 1, \\ \tau(u-1), & u > 1. \end{cases}$$

where $\tau \ge 0$ is the preset parameter. It is obvious that ℓ_{hinge} is a special case of ℓ_{pinball} when $\tau = 0$. The classification models obtained by combining the above

two loss functions with the L_2-norm penalty are referred to as HSVM [2] and PSVM [3] respectively.

Both hinge loss function and pinball loss function are nondifferentiable (at zero point) convex functions. In applications, non-differentiability restricts the choice of algorithms, while convexity makes the loss function value of outliers larger, thus greatly affects the model effect.

In order to overcome this weakness of convexity, the truncation method is a common idea, which is to redefine large values as some constant (s). For example, the truncation hinge loss function in RSVM [9] is

$$\ell(u)=\min\left\{s,\ell_{\text{hinge}}(u)\right\}=\begin{cases}s, & u\le 1-s,\\ 1-u, & 1-s<u\le 1,\\ 0, & u>1.\end{cases}$$

Such operation makes the model more stable when dealing with noisy data. However, the truncation method makes the loss function nonconvex, so it has higher requirements for the algorithms.

An important method for handling non-differentiability is to locally correct the function values near the non-differentiable points, to make the entire function differentiable. Huberization is a common correction approach that uses quadratic functions to handle non-differentiable points. For example, the Huberized hinge loss function used in HHSVM [14]:

$$\ell_H(u)=\begin{cases}0, & u>1,\\ \dfrac{(1-u)^2}{2\delta}, & 1-\delta<u\le 1,\\ 1-u-\dfrac{\delta}{2}, & u\le 1-\delta,\end{cases}$$

where $\delta>0$ is a pre-specified constant.

Based on the above methods, Zhu *et al.* [10] proposed the HTPSVM classification model:

$$\min_{w,b}\frac{1}{m}\sum_{i=1}^{m}\ell_{\text{htp}}\left(\boldsymbol{y}_i\left(w^{\mathrm{T}}\boldsymbol{x}_i+b\right)\right)+\lambda_1\|w\|_1+\frac{\lambda_2}{2}\|w\|_2^2+\frac{\lambda_3}{2}b^2.$$

where $\lambda_1,\lambda_2\ge 0$ are regularization parameters. The loss function is

$$\ell_{\text{htp}}(u)=\begin{cases}\theta_1, & u\le 1-\delta-\theta_1,\\ 1-u-\dfrac{\delta}{2}-\dfrac{(u-1+\theta_1)^2}{2\delta}, & 1-\delta-\theta_1<u\le 1-\delta,\\ 1-u-\dfrac{\delta}{2}, & 1-\theta_1<u\le 1-\delta,\\ \dfrac{(1-u)^2}{2\delta}, & 1-\delta<u\le 1,\\ \dfrac{\tau(1-u)^2}{2\delta}, & 1<u\le 1+\delta,\\ -\tau\left(1-u+\dfrac{\delta}{2}\right), & 1+\delta<u\le 1+\theta_2,\\ -\tau\left(1-u+\dfrac{\delta}{2}\right)-\dfrac{\tau(u-1-\theta_2)^2}{2\delta}, & 1+\theta_2<u\le 1+\delta+\theta_2,\\ \tau\theta_2, & u>1+\delta+\theta_2.\end{cases}$$

where $\theta_1, \theta_2 > 0$, $\delta \in [0, 1\min\{\theta_1, \theta_2\})$, $\tau \in [0,1]$ are pre-specified constants. Here we note that ℓ_{htp} is obtained by truncating the pinball loss function and then applying Huberization. Experiments validate that HTPSVM not only has better performance in terms of noise resistance compared to other methods, but also demonstrates improved time efficiency in solving the problem using the APG algorithm.

3. Robust Support Vector Machine Classifier with the Quartic Truncated Pinball Loss (QTPSVM)

Inspired by the Huberization method and HTPSVM, we replace the quadratic term in HTPSVM with a quartic function to obtain the following quartic truncated pinball loss function:

$$l_{\text{qtp}}(u) = \begin{cases} \theta_1\delta^2, & u \le 1-\delta-\theta_1, \\ -\delta^2 u - \frac{3}{4}\delta^3 + \delta^2 - \frac{(u-1+\theta_1)^4}{4\delta}, & 1-\delta-\theta_1 < u \le 1-\theta_1, \\ -\delta^2 u - \frac{3}{4}\delta^3 + \delta^2, & 1-\theta_1 < u \le 1-\delta, \\ \frac{(1-u)^4}{4\delta}, & 1-\delta < u \le 1, \\ \frac{\tau(1-u)^4}{4\delta}, & 1 < u \le 1+\delta, \\ \tau\delta^2 u - \frac{3}{4}\tau\delta^3 - \tau\delta^2, & 1+\delta < u \le 1+\theta_2, \\ \tau\delta^2 u - \frac{3}{4}\tau\delta^3 - \tau\delta^2 - \frac{\tau(u-1-\theta_2)^4}{4\delta}, & 1+\theta_2 < u \le 1+\delta+\theta_2, \\ \tau\theta_2\delta^2, & u > 1+\delta+\theta_2, \end{cases}$$

where $\theta_1, \theta_2 > 0$, $0 \le \delta \le \min\{\theta_1, \theta_2\}$, $\tau \in [0,1]$ are pre-specified constants. We notice that $\ell_{\text{qtp}}(u)$ is bounded, continuous and differentiable, but not convex.

The SVM classifier with the quartic truncated pinball loss (QTPSVM) based on the elastic net penalty is defined as:

$$\min_{w,b} \frac{1}{m}\sum_{i=1}^{m} \ell_{\text{qtp}}\left(\boldsymbol{y}_i\left(w^{\mathrm{T}}\boldsymbol{x}_i + b\right)\right) + \lambda_1\|w\|_1 + \frac{\lambda_2}{2}\|w\|_2^2 + \frac{\lambda_3}{2}b^2. \tag{3.1}$$

where $\lambda_1, \lambda_2, \lambda_3$ are regularization parameters. The advantage of QTPSVM over HTPSVM lies in the fact that, compared to the uniform change of derivatives for quadratic functions, the derivative of a quartic function changes very slowly within each segment and more rapidly at the boundaries, with relatively smaller fluctuations in function values. This makes the impact of misclassified points near the separating hyperplane on the function value and derivative smaller, thereby enhances the noise resistance of the model. However, quartic functions may lead higher requirements on algorithms, so we need a suitable method to solve this problem, which is the BAPGS algorithm we introduce below.

4. The Bregman Modified Second APG Method for Solving QTPSVM Problem

The quartic truncated pinball loss is nonconvex. However, many algorithms for nonconvex optimization problems may not perform well on quartic functions due to the usage of the proximal operator.

So to solve (3.1), we adopt the Bregman modified second accelerated proximal gradient (BAPGs) method, which is a variant of the modified second accelerated proximal gradient (APGs) method by replacing the proximal operator by the minimization involving the Bregman distance.

4.1. The BAPGs Method

The BAPGs are used to solve optimization problems in the following form:

$$\min F(x) := f(x) + P_1(x) - P_2(x), \tag{4.1}$$

where f is a differentiable function on $\mathbb{R}^d$, P_1 is proper, convex and lower semicontinuous, and P_2 is proper and convex.

Inspired by Nesterov's second acceleration method [15], Lin and Liu proposed a modified second accelerated proximal gradient (APGs) method for this type of problem. At each iteration, it depends on the resolution of the subproblem,

$$z^{k+1} = \arg\min_{z\in\mathbb{R}^d} \left\{ \left\langle \nabla f(y^k) - \xi^k, z - y^k \right\rangle + P_1(z) + \frac{\theta_k}{2} L \left\| z - z^k \right\|^2 \right\}, \tag{4.2}$$

where $\xi^k \in \partial P_2(x^k)$ is a subgradient of P_2 at x^k. Ren, Liu and Wang proved the convergence properties of this method when f is nonconvex [16]. To ensure global convergence of the iteration, both in [17] and [16], f need to be L-smooth, *i.e.*, $\left\| \nabla f(x) - \nabla f(y) \right\| \le L \left\| x - y \right\|$ for all $x, y \in \operatorname{int}\operatorname{dom} f$.

Based on the good performance of APGs in many problems in the form of (4.1) and some improvement of introducing Bregman distance into existing algorithms, Wang, Liu and Ren combine Bregman distance with APGs to construct a new algorithm [11].

Definition (Bregman Distances [18]) Suppose that $h: \mathbb{R}^d \to (-\infty, +\infty]$ is a proper, continuously differentiable, convex function, the following function $D_h: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ is called the Bregman distance corresponding to h:

$$D_h(x, y) := h(x) - h(y) - \left\langle \nabla h(y), x - y \right\rangle.$$

The BAPGs algorithm are shown as follows

Algorithm 1 BAPGs

Initialization: $x^0 = z^0 \in \operatorname{dom} P_1$

For $k = 0, 1, \ldots,$ let any $\xi^k \in \partial P_2(x^k)$ and do

$$\begin{aligned} y^k &= (1 - \theta_k)x^k + \theta_k z^k, \\ z^{k+1} &= \arg\min_{z\in\mathbb{R}^d} \left\{ \left\langle \nabla f(y^k) - \xi^k, z - y^k \right\rangle + P_1(z) + \sigma L \theta_k D_h(z, z^k) \right\}, \\ x^{k+1} &= (1 - \theta_k)x^k + \theta_k z^{k+1}. \end{aligned}$$

End for

4.2. Settings for the BAPGs Method on QTPSVM

In order to apply BAPGs to the QTPSVM problems, we first organize problem (3.1) into the form of (4.1), (we always use x to denote (w,b) in the details of the algorithm)

$$f(x) = f_1(x) - f_3(x),$$
$$P_1(x) = P_1(w,b) = \lambda_1 \|w\|_1 + \frac{\lambda_2}{2}\|w\|_2^2 + \frac{\lambda_3}{2}b^2,$$
$$P_2(x) = f_2(x),$$

where $f_j(x) = f_j(w,b) = \frac{1}{m}\sum_{i=1}^{m} l_j\left(y_i\left(b + w^{\mathrm{T}}x_i\right)\right)$, $j = 1,2,3$,

$$l_1(u) = \begin{cases} -\delta^2 u - \frac{3}{4}\delta^3 + \delta^2, & u \le 1-\delta, \\ \frac{(1-u)^4}{4\delta}, & 1-\delta < u \le 1, \\ \frac{\tau(1-u)^4}{4\delta}, & 1 < u \le 1+\delta, \\ \tau\left(\delta^2 u - \frac{3}{4}\delta^3 - \delta^2\right), & u > 1+\delta, \end{cases}$$

$$l_2(u) = \begin{cases} -\delta^2 u - \frac{3}{4}\delta^3 + \delta^2 - \theta_1\delta^2, & u \le 1-\delta-\theta_1, \\ \frac{(u-1+\theta_1)^4}{4\delta}, & 1-\delta-\theta_1 < u \le 1-\theta_1, \\ 0, & u > 1-\theta_1, \end{cases}$$

$$l_3(u) = \begin{cases} 0, & u \le 1+\theta_2, \\ \frac{\tau(u-1-\theta_2)^4}{4\delta}, & 1+\theta_2 < u \le 1+\delta+\theta_2, \\ \tau\left(\delta^2 u - \frac{3}{4}\delta^3 - \delta^2 - \theta_2\delta^2\right), & u > 1+\delta+\theta_2. \end{cases}$$

According to the common setting of similar problems [19], we take $h(x) = \frac{1}{4}\|x\|_2^4$. In theory, parameter L should ensure that $Lh \pm f$ is a convex function, namely for any $x = (w,b) \in \mathbb{R}^n \times \mathbb{R}$,

$$L\nabla^2 h(x) \pm \nabla^2 f(x) \succeq 0,$$

Noting that $\nabla^2 h(x) = \|x\|^2 I + 2xx^{\mathrm{T}}$, $\nabla^2 f(x) = \frac{1}{m}\sum_{i=1}^{m} Q_i$, and

$$Q_i = \begin{cases} \frac{3}{\delta}(u_i - 1)^2 Q_i, & 1-\delta < u_i \le 1, \\ \frac{3\tau}{\delta}(u_i - 1)^2 Q_i, & 1 < u \le 1+\delta, \\ -\frac{3\tau}{\delta}(u_i - 1 - \theta_2)^2 Q_i, & 1+\theta_2 < u \le 1+\delta+\theta_2, \\ 0, & \text{else.} \end{cases}$$

$Q_i = \left(\boldsymbol{y}_i \boldsymbol{x}_i^{\mathrm{T}}, \boldsymbol{y}_i\right)^{\mathrm{T}} \left(\boldsymbol{y}_i \boldsymbol{x}_i^{\mathrm{T}}, \boldsymbol{y}_i\right)$, $u_i = \boldsymbol{y}_i \left(w^{\mathrm{T}} \boldsymbol{x}_i + b\right)$. Here we see that precisely calculating the range of L is very difficult. However, in actual algorithms, we only need $Lh \pm f$ to be convex near the optimal point, and in the experiments, the variations of $\nabla^2 h(x)$ and $\nabla^2 f(x)$ near the optimal point are not significant. Therefore, we can approximately estimate L at the initial point (if the initial point is zero, we set $L = 1$), and adjust it at x^N (where N is introduced next), then we restart the algorithm with new L from the original initial point.

For the sequence $\{\theta_k\}$, we let $\theta_0 = 1$, and for a positive integer N, let

$$\theta_k := \frac{\sqrt{\theta_{k-1}^4 + 4\theta_{k-1}^2} - \theta_{k-1}^2}{2}$$

when $1 \le k \le N$, and $\theta_k = \theta_N$ for $k > N$. The selection of N is flexibly determined in the algorithm according to the judgment method in [16].

Finally, the following result is used to obtain z^{k+1}:

Proposition For $p \in \mathbb{R}^d, \lambda > 0$, let $S_\lambda(p)$ be the soft-thresholding operator, *i.e.*, $S_\lambda(p) = \max\{|p| - \lambda, 0\} \operatorname{sgn}(p)$. Let

$$z^+ := \operatorname{argmin}\left\{\lambda \|z\|_1 + \langle p, z\rangle + c_1 \|z\|_2^4 + c_2 \|z\|_2^2\right\},$$

where c_1, c_2 are positive constants. We have $z^+ = 0$ when $S_\lambda(p) = 0$, and $z^+ = -\dfrac{S_\lambda(p)}{4c_1 \left\|z^+\right\|_2^2 + 2c_2}$ otherwise, where $\left\|z^+\right\|_2$ is such that $\left\|S_\lambda(p)\right\|_2 = 4c_1 \left\|z^+\right\|_2^3 + 2c_2 \left\|z^+\right\|_2$.

Proof By the first order optimality condition for convex problems, we have

$$0 \in \lambda \partial \|\cdot\|_1 \left(z^+\right) + p + 4c_1 \left\|z^+\right\|_2^2 z^+ + 2c_2 z^+.$$

Let $v = -4c_1 \left\|z^+\right\|_2^2 z^+ - 2c_2 z^+$, we have $0 \in \lambda \partial \|\cdot\|_1 (v) - p + v$, thus $v = S_\lambda(p)$. Noting that $\|v\|_2 = 4c_1 \left\|z^+\right\|_2^3 + 2c_2 \left\|z^+\right\|_2$, the conclusion is proved.

5. Numerical Experiments

To evaluate the performance of QTPSVM, we compare it with HTPSVM on real datasets. The HTPSVM is solved by the proximal gradient algorithm (APG) as in [20]. The experimental datasets are all taken from the UCI Machine Learning Repository [21]. Both models start from the zero vector for all experiments. The algorithm stops if both the following two conditions are met:

$$\frac{F_{k-1} - F_k}{1 + F_{k-1}} \le \mathrm{tol}, \quad \frac{\left\|x^k - x^{k-1}\right\|}{1 + \left\|x^{k-1}\right\|} \le \mathrm{tol},$$

where $x^k = \left(w^k, b^k\right)$, $F_k = F\left(x^k\right)$. The stopping tolerance $\mathrm{tol} = 10^{-6}$.

As is well known, the choice of parameters plays a crucial role in the effectiveness of machine learning [22]-[24]. The parameters for the APG are set as in [10]. For QTPSVM and HTPSVM, we take $\lambda_1 = 5 \times 10^{-4}$, and $\tau = 0.5$, and fix $\lambda_2 = \lambda_3 = 1$ because the algorithm seems not sensitive to λ_2 and λ_3 [10]. In

HTPSVM, θ_1, θ_2, and δ are set according to the original paper, that is, they are tuned through 10-fold cross-validation from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. In QTPSVM, we found by tenfold cross-validation on the same parameter combination set that experiments perform best and most efficiently on most dataset. Other combinations are not only less effective in most cases but also time-consuming, so we fix this parameter combination. After determining the parameter combination, for each dataset, we randomly divide it into training and test sets 50 times; for each division, 80% of the dataset is used for training, and the remaining 20% is used for testing.

All numerical experiments were conducted on an Intel(R) Core(TM) i5-8265U processor with 8.00 GB memory. The operating system is Windows 10, and the version of Matlab used is R2021a.

We compare QTPSVM with HTPSVM. To further study the robustness of QTPSVM, we randomly flip the labels of the training set, with flip levels of 0%, 5%, and 10%. The classification accuracy on the test set (averaged over 50 repetitions) is recorded in Table 1. We also record the average running time for each method (in seconds) in Table 2. To further evaluate the performance of the classifiers, the area under the ROC curve (AUC) is calculated and recorded in Table 3.

Through Tables 1-3, we can observe that the average running time of QTPSVM is less than that of HTPSVM. In terms of classification accuracy, QTPSVM has higher accuracy and AUC on many datasets compared to HTPSVM. This improvement in performance remains stable even when different levels of label flipping are applied.

Table 1. Classification accuracy on real data sets (%).

	0%		5%		10%	
	QTPSVM	HTPSVM	QTPSVM	HTPSVM	QTPSVM	HTPSVM
Rocks	**77.02**	75.17	**78.05**	76.34	**76.59**	74.83
Bupa	**60.14**	57.28	**57.01**	54.41	**59.74**	56.78
Fourclass	75.28	**75.64**	75.57	**76.23**	75.29	**75.77**
German	**75.59**	74.42	**75.67**	74.33	**75.56**	74.63
Glass	**92.62**	91.67	**91.14**	90.05	**91.95**	90.81
Haperman	**73.41**	72.30	**73.28**	72.69	**74.49**	73.80
Heart	84.81	**85.52**	83.78	**84.41**	84.59	**84.93**
Hepatitis	85.57	**85.79**	84.21	**84.64**	83.29	**83.50**
Liver	**67.07**	66.32	65.30	**65.36**	**64.61**	63.62
Raisin	**85.51**	84.38	**84.01**	83.17	**85.84**	84.86
Tic	**66.92**	65.31	**67.35**	65.21	**67.78**	66.31

Table 2. Computation time on test data of real data sets (in seconds).

	0%		5%		10%	
	QTPSVM	HTPSVM	QTPSVM	HTPSVM	QTPSVM	HTPSVM
Rocks	**0.0278**	0.5550	**0.0215**	0.6410	**0.0226**	0.4948
Bupa	**0.0150**	0.4181	**0.0137**	0.5011	**0.0131**	0.5011
Fourclass	**0.0281**	1.3818	**0.0283**	1.5580	**0.0265**	1.5142
German	**0.0331**	0.6596	**0.0320**	0.6879	**0.0285**	0.8484
Glass	**0.0150**	0.4931	**0.0151**	0.4548	**0.0131**	0.5007
Haperman	**0.0180**	0.6065	**0.0181**	0.5975	**0.0163**	0.6039
Heart	0.0145	**0.0100**	0.0164	**0.0081**	**0.0136**	0.0153
Hepatitis	**0.0165**	0.0355	**0.0163**	0.0366	**0.0148**	0.0666
Liver	**0.0176**	0.5612	**0.01195**	0.4868	**0.0176**	0.6065
Raisin	0.0330	**0.0204**	0.0309	**0.0180**	0.0299	**0.0176**
Tic	**0.0310**	1.6001	**0.0294**	1.6136	**0.0303**	1.5903

Table 3. AUC of four classification models on real datasets.

	0%		5%		10%	
	QTPSVM	HTPSVM	QTPSVM	HTPSVM	QTPSVM	HTPSVM
Rocks	**0.8539**	0.8079	**0.8721**	0.8142	**0.8440**	0.8012
Bupa	**0.6328**	0.6106	**0.6375**	0.5969	**0.6378**	0.6229
Fourclass	**0.8308**	0.8299	**0.8341**	0.7789	**0.7795**	0.7755
German	**0.7899**	0.7843	**0.7849**	0.7789	**0.7795**	0.7755
Glass	0.9706	**0.9745**	0.9637	**0.9652**	0.9608	**0.9677**
Haperman	**0.6881**	0.6863	0.6774	**0.6796**	**0.6850**	0.6713
Heart	**0.9183**	0.9176	0.9042	**0.9054**	**0.9114**	0.9104
Hepatitis	0.8954	**0.8979**	0.8845	**0.8861**	**0.8674**	0.8663
Liver	**0.6679**	0.6635	**0.6533**	0.6487	**0.6557**	0.6523
Raisin	0.9258	**0.9266**	0.9171	**0.9182**	0.9278	**0.9285**
Tic	**0.6172**	0.6130	**0.6265**	0.6194	**0.6116**	0.6064

6. Conclusion

In this paper we propose a new robust support vector machine classification model QTPSVM, based on a quartic truncated pinball loss function, and solve it using the BAPGs method. By comparing the numerical results of QTPSVM and HTPSVM in real data, it is shown that this new classifier is effective and stable to noise. However, QTPSVM involves multiple parameters, and parameter selection requires cross-validation tuning, which increases the complexity and computational cost of model training. Therefore, QTPSVM may not be the most efficient choice for very large-scale datasets.

Use of Generative-AI Tools Declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflicts of Interest

The authors declare that there is no conflict of interest in this manuscript.

References

[1] Vapnik, V.N. (1995) The Nature of Statistical Learning Theory. Springer-Verlag. https://doi.org/10.1007/978-1-4757-2440-0

[2] Bishop, C.M. (2006) Pattern Recognition and Machine Learning. Springer.

[3] Huang, X.L., Shi, L. and Suykens, J.A.K. (2014) Support Vector Machine Classifier with Pinball Loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**, 984-997. https://doi.org/10.1109/tpami.2013.178

[4] Rastogi (nee. Khemchandani), R., Pal, A. and Chandra, S. (2018) Generalized Pinball Loss SVMs. *Neurocomputing*, **322**, 151-165. https://doi.org/10.1016/j.neucom.2018.08.079

[5] Chong, E.K.P. and Zak, S.H. (2013) An Introduction to Optimization. 4th Edition, Wiley.

[6] Luo, J., Qiao, H. and Zhang, B. (2021) Learning with Smooth Hinge Losses. *Neurocomputing*, **463**, 379-387. https://doi.org/10.1016/j.neucom.2021.08.060

[7] Zhu, W., Song, Y. and Xiao, Y. (2020) Support Vector Machine Classifier with HUberized Pinball Loss. *Engineering Applications of Artificial Intelligence*, **91**, Article ID: 103635. https://doi.org/10.1016/j.engappai.2020.103635

[8] Makmuang, D., Ratiphaphongthon, W. and Wangkeeree, R. (2023) Smooth Support Vector Machine with Generalized Pinball Loss for Pattern Classification. *The Journal of Supercomputing*, **79**, 11684-11706. https://doi.org/10.1007/s11227-023-05082-w

[9] Wu, Y. and Liu, Y. (2007) Robust Truncated Hinge Loss Support Vector Machines. *Journal of the American Statistical Association*, **102**, 974-983. https://doi.org/10.1198/016214507000000617

[10] Zhu, W., Song, Y. and Xiao, Y. (2022) Robust Support Vector Machine Classifier with Truncated Loss Function by Gradient Algorithm. *Computers & Industrial Engineering*, **172**, Article ID: 108630. https://doi.org/10.1016/j.cie.2022.108630

[11] Wang, L.M., Liu, Z.Y. and Liu, C.G. (2024) The Bregman Modified Second APG Method for DC Optimization Problems.

[12] Becker, N., Toedt, G., Lichter, P. and Benner, A. (2011) Elastic SCAD as a Novel Penalization Method for SVM Classification Tasks in High-Dimensional Data. *BMC Bioinformatics*, **12**, Article No. 138. https://doi.org/10.1186/1471-2105-12-138

[13] Zou, H. and Hastie, T. (2005) Addendum: Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **67**, 768-768. https://doi.org/10.1111/j.1467-9868.2005.00527.x

[14] Wang, L., Zhu, J. and Zou, H. (2008) Hybrid Huberized Support Vector Machines for Microarray Classification and Gene Selection. *Bioinformatics*, **24**, 412-419. https://doi.org/10.1093/bioinformatics/btm579

[15] Nesterov, Y. (2013) Introductory Lectures on Convex Optimization: A Basic Course. Springer Science & Business Media.

[16] Ren, K., Liu, C. and Wang, L. (2024) The Modified Second APG Method for a Class of Nonconvex Nonsmooth Problems. *Optimization Letters*, **19**, 747-770. https://doi.org/10.1007/s11590-024-02132-x

[17] Lin, D. and Liu, C. (2018) The Modified Second APG Method for DC Optimization Problems. *Optimization Letters*, **13**, 805-824. https://doi.org/10.1007/s11590-018-1280-8

[18] Bregman, L.M. (1967) The Relaxation Method of Finding the Common Point of Convex Sets and Its Application to the Solution of Problems in Convex Programming. *USSR Computational Mathematics and Mathematical Physics*, **7**, 200-217. https://doi.org/10.1016/0041-5553(67)90040-7

[19] Takahashi, S., Fukuda, M. and Tanaka, M. (2022) New Bregman Proximal Type Algorithms for Solving DC Optimization Problems. *Computational Optimization and Applications*, **83**, 893-931. https://doi.org/10.1007/s10589-022-00411-w

[20] Li, Q.W., Zhou, Y., Liang, Y.B. and Varshney, P.K. (2017) Convergence Analysis of Proximal Gradient with Momentum for Nonconvex Optimization. *Proceedings of the 34th International Conference on Machine Learning*, Sydney, 6-11 August 2017, 2111-2119.

[21] Dua, D. and Graff, C. (2019) UCI Machine Learning Repository. University of California, School of Information and Computer Science.

[22] Braga, I., do Carmo, L.P., Benatti, C.C. and Monard, M.C. (2013) A Note on Parameter Selection for Support Vector Machines. In: Castro, F., Gelbukh, A. and González, M., Eds., *Advances in Soft Computing and Its Applications*. *MICAI* 2013, Springer, 233-244. https://doi.org/10.1007/978-3-642-45111-9_21

[23] Huang, H., Wang, Z. and Chung, W. (2019) Efficient Parameter Selection for Support Vector Machines. *Enterprise Information Systems*, **13**, 916-932. https://doi.org/10.1080/17517575.2019.1592233

[24] Wang, H., Xu, D. and Martinez, A. (2018) Parameter Selection Method for Support Vector Machine Based on Adaptive Fusion of Multiple Kernel Functions and Its Application in Fault Diagnosis. *Neural Computing and Applications*, **32**, 183-193. https://doi.org/10.1007/s00521-018-3792-7