entropy

MDPI

Article

Robust Support Vector Data Description with Truncated Loss Function for Outliers Depression

Huakun Chen, Yongxi Lyu *, Jingping Shi and Weiguo Zhang

Department of Automatic Control, Northwestern Polytechnical University, Xi'an 710072, China; chenhuakun@mail.nwpu.edu.cn (H.C.); shijingping@nwpu.edu.cn (J.S.); zhangwg@nwpu.edu.cn (W.Z.)
* Correspondence: yongxilyu@nwpu.edu.cn

Abstract: Support vector data description (SVDD) is widely regarded as an effective technique for addressing anomaly detection problems. However, its performance can significantly deteriorate when the training data are affected by outliers or mislabeled observations. This study introduces a universal truncated loss function framework into the SVDD model to enhance its robustness and employs the fast alternating direction method of multipliers (ADMM) algorithm to solve various truncated loss functions. Moreover, the convergence of the fast ADMM algorithm is analyzed theoretically. Within this framework, we developed the truncated generalized ramp, truncated binary cross entropy, and truncated linear exponential loss functions for SVDD. We conducted extensive experiments on synthetic and real-world datasets to validate the effectiveness of these three SVDD models in handling data with different noise levels, demonstrating their superior robustness and generalization capabilities compared to other SVDD models.

Keywords: SVDD; truncated loss function; fast ADMM; proximal operators; anomaly detection; truncated binary cross entropy loss function; truncated linear exponential loss function

check for updates

Citation: Chen, H.; Lyu, Y.; Shi, J.; Zhang, W. Robust Support Vector Data Description with Truncated Loss Function for Outliers Depression. *Entropy* **2024**, *26*, 628. https://doi.org/10.3390/e26080628

Academic Editor: Kenji Yamanishi

Received: 11 June 2024
Revised: 19 July 2024
Accepted: 23 July 2024
Published: 25 July 2024

1. Introduction

Anomaly detection refers to the identification of data points in a dataset that deviate from normal behavior. These deviations are known as anomalies or outliers in various application domains. This mode of detection is extensively used in real-world settings, including credit card detection, insurance detection, cybersecurity intrusion detection, error detection in security systems, and military activity monitoring [1,2]. However, the acquisition of anomaly data in practical applications, such as medical diagnostics, machine malfunction detection, and circuit quality inspection, is expensive [3,4]. Consequently, there is significant interest in one-class classification (OCC) problems, where training samples only include normal data (also known as target data) or normal data with a small number of anomalies (also referred to as non-target data) [5–7]. In this context, it is important to define the following terms:

- Normal data: Normal data refers to data points that conform to the characteristics and behavior patterns of the majority of data points within a dataset. They represent the normal operating state of a system or process.
- Anomalies: Anomalies are data points that significantly deviate from normal patterns and usually reflect actual problems or critical events in the system.
- Outliers: Outliers are data points that are significantly different from other data points in the dataset, which may be due to natural fluctuations, special circumstances, or noise.
- Noise: Noise refers to irregular, random errors or fluctuations, usually caused by measurement errors or data entry mistakes, and does not reflect the actual state of the system.

Support vector data description (SVDD) is a method extensively used for one-class classifications (OCCs) [8]. The core idea of SVDD is to construct a hyper-sphere of minimal volume that encompasses all (or most) of the target class samples. Data points inside the hyper-sphere are considered normal, while those outside are considered anomalies. SVDD can be easily integrated with popular kernel methods or some deep neural network models [7], making it highly scalable and flexible. Due to these attractive features, SVDD has garnered significant attention and has been extensively developed. SVDD is regarded as an effective and excellent technique for anomaly detection problems; however, it remains sensitive to outliers and noise present in training datasets. In real-world scenarios, various issues, such as instrument failure, formatting errors, and unrepresentative sampling, result in datasets with anomalies, which degrade the performance of the SVDD [9,10].

The existing methods for mitigating the impact of noise are typically categorized as follows:

In order to reduce the impact of outliers on the OCC method, researchers have attempted to remove outliers through data preprocessing methods. Stanley Fong used methods such as cluster analysis to remove anomalies from the training set to achieve a robust classifier [11]. Breunig et al. tried to assign an outlier score to each sample in the dataset by estimating its local density, known as LOF [12]. Zheng et al. used LOF to filter raw samples and remove outliers [13]. Khan et al. [14] and Andreou and Karathanassi [15] calculated the interquartile range (IQR) of training samples, which provides a method for indicating a boundary beyond which samples are marked as outliers and removed. Clustering (k-means, DBSCAN) or LOF methods can significantly reduce noise points in data preprocessing, thereby improving the quality and effectiveness of subsequent model training. For datasets with obvious noise and significant distribution characteristics, preprocessing methods can be very effective in enhancing model performance. However, preprocessing methods have several issues: they require additional computational resources and time, especially with large datasets, potentially making the preprocessing step time-consuming. Moreover, there is a risk of overfitting and inadvertently deleting useful normal data points, impacting the model's ability to accurately detect anomalies. Additionally, these preprocessing methods are sensitive to parameter settings, necessitating careful selection to achieve satisfactory results.

Zhao et al. proposed the dynamic radius SVDD method that accounts for hyper-sphere radius information and the existing data distribution [16,17]. This approach achieves a more flexible decision boundary by assigning different radii to different samples. Moreover, the framework for the dynamic radius SVDD method is based on the traditional SVDD framework. However, if the traditional SVDD has not undergone adequate training, the good performance of these dynamic approaches will be difficult to guarantee.

Density-weighted SVDD, position-weighted SVDD, Stahel–Donoho outlier-weighted SVDD, global plus local joint-weighted SVDD, and confidence-weighted SVDD are examples of weighted methods [18–24]. These methods assign smaller weights to sparse data, which are commonly outliers, thus excluding them from the sphere. These methods balance the target class data and outliers in the training phase, thus enhancing the classification performance, especially when the data are contaminated by outliers. However, when the number of outliers in the dataset increases and they form sparse clusters, the number of outliers might surpass that of normal samples. In such cases, weighted methods assign higher weights to the outliers and lower weights to the normal samples, leading to decreased algorithm performance.

The convex property of the hinge loss function of the SVDD algorithm makes it sensitive to outliers. To address this issue, Xing et al. proposed a new robust least squares one-class support vector machine (OCSVM) that employs a bounded, non-convex entropic loss function, instead of the unbounded convex quadratic loss function used in traditional least squares OCSVM [25]. The non-convex nature of the ramp loss function makes this model more robust than the traditional OCSVM [26]. Tian et al. introduced the ramp loss function to the traditional OCSVM to create the Ramp-OCSVM model [27], and the non-convex nature of this model makes it more robust than the traditional OCSVM. Xing et al. enhanced the robustness of the OCSVM by introducing a re-scaled hinge loss function [28]. Additionally, Zhong et al. proposed a new robust SVDD method, called pinball loss SVDD [29], to perform OCC tasks when the data are contaminated by outliers. The pinball loss function ensures minimal dispersion at the center of the sphere, thus creating a tighter decision boundary. Recently, Zheng introduced a mixed exponential loss function to the design of the SVDD model, enhancing its robustness and making its implementation easier [30].

Extensive research has shown that, as a result of unbounded convex loss functions being sensitive to anomalies, loss functions with boundedness or bounded influence functions are more robust to the influence of outliers. To address this issue, researchers introduced an upper limit to unbounded loss functions, effectively preventing them from increasing beyond a certain point. The truncated loss function thus makes the SVDD model more robust. The advantages of truncated loss functions include:

- Robustness to noise: Truncated loss functions can enhance the model's robustness and stability by limiting the impact of outliers without removing data points.
- Reduction of error propagation: In anomaly detection tasks, outliers may significantly contribute to the loss function, leading to error propagation and model instability. Truncated loss functions can effectively reduce error propagation caused by outliers, thereby improving overall model performance.
- Generalization ability: Using truncated loss functions can prevent the model from overfitting to outliers, enhancing the model's generalization ability. Truncated loss functions are well-suited for various types of datasets and noise conditions, particularly when noise is not obvious or easily detectable.

However, robust SVDD algorithms still face considerable challenges in the research. They are designed to address specific types of losses and lack an appropriate framework for constructing robust loss functions. Thus, researchers are required to learn how to use different algorithms and modify loss functions before use. Since truncated loss functions are often non-differentiable, methods such as the difference of convex algorithm (DCA) [31] and concave–convex procedures (CCCPs) [32,33] are commonly employed to provide solutions. For some truncated loss functions, the DCA cannot ensure straightforward decompositions or the direct use of comprehensive convex toolboxes, potentially increasing development and maintenance costs [34]. At present, no unified framework exists in the literature for the design of robust loss functions or a unified optimization algorithm. Therefore, even though this is challenging, providing a new bounded strategy for the SVDD model is crucial, with the potential for developing more efficient and universally applicable solutions.

In response to the several issues previously outlined, this study proposes a universal framework for the truncated loss functions of the SVDD model. To address and solve the non-differentiable, non-convex optimization problem introduced by the truncated loss function, we employ the fast ADMM algorithm. Our contributions to this field of study are as follows:

- We define a universal truncated loss framework that smoothly and adaptively binds loss functions, while preserving their symmetry and sparsity.
- To solve different truncated loss functions, we propose the use of a unified proximal operator algorithm.
- We introduce a fast ADMM algorithm to handle any truncated loss function within a unified scheme.
- We implement the proposed robust SVDD model for various datasets with different noise intensities. The experimental results for real datasets show that the proposed model exhibits superior resistance to outliers and noise compared to more traditional methods.

The remainder of this paper is organized as follows:

Section 2: We review related support vector data description (SVDD) models, providing a foundational understanding of the existing methodologies and their limitations.

Section 3: We propose a general framework for truncated loss functions. Within this framework, we examine the representative loss functions' proximal operators and present a universal algorithm for solving these proximal operators.

Section 4: This section introduces the SVDD model that utilizes the truncated loss function, detailing its structure and theoretical framework.

Section 5: A new algorithm for solving the SVDD model with truncated loss functions is presented. This section also includes an analysis of the algorithm's convergence properties, ensuring that the method is both robust and reliable.

Section 6: Numerical experiments and parameter analysis are conducted to validate the effectiveness of the proposed model. This section provides empirical evidence of the model's performance across various datasets and noise scenarios.

Section 7: The conclusion summarizes the findings and contributions of the study, and discusses potential future research directions.

2. Related Works

SVDD has been widely applied in anomaly detection, and numerous learning algorithms based on SVDD have been proposed. In this section, we provide a brief overview of these algorithms.

2.1. SVDD

The goal of SVDD is to discover a hyper-sphere that encompasses target samples, while excluding non-target samples located outside it [8]. The objective function of SVDD is represented by the following equation:

$$\begin{aligned} &\min_{R,\mu,\xi_i} R^2 + C\sum_i \xi_i \\ &\text{s.t } \|\varphi(x_i) - \mu\|^2 \le R^2 + \xi_i,\ \xi_i > 0,\ \forall i \end{aligned} \tag{1}$$

where μ and R represent the radius and center of the hyper-sphere, C is a regularization parameter, and ξ_i is a slack variable. By using Lagrange multipliers and incorporating the constraints into the objective function, the dual problem of Equation (1) can be expressed as:

$$\begin{aligned} &\max \sum_i a_i(x_i \cdot x_i) - \sum_{i,j} a_i a_j (x_i \cdot x_j) \\ &\text{s.t } \sum_i a_i = 1,\ 0 \le a_i \le C,\ \forall i \end{aligned} \tag{2}$$

where α_i is the Lagrange multiplier, the optimization problem in Equation (2) is a standard quadratic programming problem, and α_i can be obtained using quadratic programming algorithms. μ and R can be calculated using the following equation:

$$\mu = \sum_{i=1}^{n} \alpha_i \varphi(x_i) \tag{3}$$

$$R^2 = (x_s \cdot x_s) - 2\sum_i \alpha_i (x_s \cdot x_i) + \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \tag{4}$$

where x_s represents support vectors. The decision function for the test sample z is presented as follows:

$$f(z) = sign\left(R^2 - \|z - \mu\|^2\right) = sign\left\{R^2 - (z \cdot z) + 2\sum_i a_i (z \cdot x_i) - \sum_{i,j} a_i a_j (x_i \cdot x_j)\right\} \tag{5}$$

If $f(z) \le 0$, then sample z belongs to the target class; otherwise, z is considered a non-target class sample.

2.2. Robust SVDD Variants

Due to its sensitivity to outliers, the classification performance of SVDD significantly deteriorates when the data are contaminated. Thus, to enhance the robustness of SVDD, various improved SVDD methods have been proposed over the past decades.

2.2.1. Weighted SVDDs

One common approach is the use of weighted SVDD methods, where different slack variables are assigned different weights [18–24]. Although the specific methods for weight distribution vary, they can generally be represented in a unified form:

$$\begin{aligned} &\min_{R,\mu,\xi_i} R^2 + C\sum_i w_i \xi_i \\ &\text{s.t } \|\varphi(x_i) - \mu\|^2 \le R^2 + \xi_i,\ \xi_i > 0,\ \forall i \end{aligned} \tag{6}$$

where $\{w_i\}_{i=1}^n$ represents pre-calculated weights. The density weighting method permits these weights to be calculated as follows [20]:

$$w_i = 1 - \frac{d\left(x_i, x_i^k\right)}{\max\left\{d\left(x_1, x_1^k\right), \cdots, d\left(x_n, x_n^k\right)\right\}} \tag{7}$$

where x_i^k denotes the k-nearest neighbor value of x_i, and $d\left(x_i, x_i^k\right)$ denotes the Euclidean distance between x_i and x_i^k. Due to outliers typically being located in relatively low-density areas, the distance to their neighboring samples is greater when compared to normal samples, which results in their smaller weights. The R-SVDD algorithm constructs weights by introducing a local density to each data point based on truncated distances [19].

$$\rho_i = \sum_j \exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right) \tag{8}$$

where d_{ij} denotes the distance between x_i and x_j, and d_c is the truncation distance. The calculation of the weight function is as follows:

$$w(x_i) = \frac{\rho(x_i)}{\max\{\rho(x_1), \cdots, \rho(x_n)\}} \tag{9}$$

Other methods for calculating can refer to [21–24]. The dual problem of Equation (6) is represented by the following equation:

$$\begin{aligned} &\max \sum_i \alpha_i (x_i \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \\ &\text{s.t} \sum_i \alpha_i = 1,\ 0 \le \alpha_i \le w_i C,\ \forall i \end{aligned} \tag{10}$$

From this equation, it can be observed that each Lagrange multiplier has an upper limit, denoted as $\alpha_i \le w_i C$.

2.2.2. Pinball Loss SVDD

The pinball loss SVDD (Pin-SVDD) modifies the hinge loss SVDD by replacing its loss function with the pinball loss function to create an optimized problem formulation [29]. This modification enables a more robust handling of outliers by adjusting the sensitivity toward deviations, depending on their direction relative to the decision boundary.

$$\begin{aligned} &\min_{R,\mu,\xi_i} R^2 + C\sum_i \xi_i \\ &\text{s.t } \|\varphi(x_i) - \mu\|^2 \le (1-\tau)R^2 + \xi_i, \\ &\quad \|\varphi(x_i) - \mu\|^2 \le \frac{1}{\tau}\xi_i, \forall i \end{aligned} \tag{11}$$

where $0 < \tau \le 1$ is a constant. With the use of the Lagrange multipliers method, the dual optimization problem in Equation (11) can be expressed as follows:

$$\begin{aligned} &\max \sum_i \alpha_i \left(g_1(x_i \cdot x_i) + g_2 \sum_{j=1}^{n} (x_i \cdot x_j) \right) + h\sum_{i,j} \alpha_i\alpha_j (x_i \cdot x_j) \\ &\text{s.t } \sum_i \alpha_i = \frac{1}{1-\tau},\ 0 \le \alpha_i \le C,\ \forall i \end{aligned} \tag{12}$$

where $h = -\frac{(1-\tau)^2}{\tau n C+1}$, $g_1 = 1-\tau$, and $g_2 = -\frac{2(1-\tau)\tau c}{\tau n c+1}$. Zhong demonstrated that the use of a pinball to minimize scattering at the center of the sphere enhances the robustness of the developed model.

2.2.3. SVDD with Mixed Exponential Loss Function

Zheng used a mixed exponential loss function to design a classification model, and the optimization problem of the method is presented by the following equation [30]:

$$\begin{aligned} &\min_{R,\mu,\xi_i} R^2 + C\sum_i \rho(\xi_i) \\ &\text{s.t } \|\varphi(x_i) - \mu\|^2 \le R^2 + \xi_i,\ \xi_i > 0,\ \forall i \end{aligned} \tag{13}$$

where $\rho(\xi_i)$ is a mixed exponential function of ξ_i, expressed as follows:

$$\rho(\xi_i) = \lambda \exp(-\tau_1\xi_i) + (1-\lambda)\exp(-\tau_2\xi_i) \tag{14}$$

where $\tau_1 > 0$ and $\tau_2 > 0$ are two scale parameters, and $1 \ge \lambda \ge 0$ is a mixture parameter used to balance the contributions of the two exponential functions. The mixed exponential loss function highlights the importance of samples involved in the target class while reducing the influence of samples that are outliers. This approach significantly enhances the robustness of the SVDD model. This loss function achieves more accurate and stable anomaly detection results in various settings by preserving the integrity of the target class and diminishing the effect of potential anomalies.

3. Truncated Loss Function

SVDD models with unbounded loss functions can achieve satisfactory results when addressing scenarios lacking noise. However, the continual growth of these loss functions results in the collapse of the model when it is subjected to noise. Therefore, truncating the SVDD model's loss function makes it more robust. The general definition of a truncated loss function is as follows:

$$L(u,\delta) = \phi(u) - (\phi(u) - \delta)_+ \tag{15}$$

where δ is a constant, and $\phi(u)$ is an unbounded loss function, such that, when $u \le 0$, $\phi(u) = 0$. Since $\phi(u)$ is an abstract function, a general form of the truncated loss function includes several loss functions. The three specific truncated loss functions we created in our study are present as follows:

1. Truncated generalized ramp loss function: $L_{\phi_R}(u,\delta) = \min\{0, \max(\delta, \phi_R(u))\}$, where $\phi_R(u) = \frac{u}{v}$, $u, v > 0$.
2. Truncated binary cross entropy loss function: $L_{\phi_f}(u,\delta) = \min\left\{0, \max\left(\delta, \phi_f(u)\right)\right\}$, where $\phi_f(u) = \log\left(1 + \frac{u}{\theta}\right)$, $u, \theta > 0$.

3. Truncated linear exponential loss function: $L_{\phi_L}(u,\delta) = \min\{0, \max(\delta, \phi_L(u))\}$, where $\phi_L(u) = \exp(ayu) - ayu - 1$, $u, a > 0$, and $y = \pm 1$.

Assuming the truncation point $u = \delta'$, the mathematical properties of the three truncated loss functions presented above can be summarized as follows:

1. For samples with $u \leq 0$, the loss value is 0; for samples with $u > \delta'$, the loss value is δ. Thus, the general truncated loss function exhibits sparsity and robustness to outliers.
2. $L_{\phi_R}(u,\delta)$ and $L_{\phi_M}(u,\delta)$ are truncated concave loss functions, which are non-differentiable at $u = 0, \delta'$. $L_{\phi_L}(u,\delta)$ is a truncated convex loss function, which is non-differentiable at $u = \delta'$ and differentiable at u= 0.
3. $L_{\phi_R}(u,\delta)$ and $L_{\phi_f}(u,\delta)$ exhibit explicit expressions for the proximal operators, while $L_{\phi_L}(u,\delta)$ does not.

In the next section, we provide explicit expressions for the proximal operators of $L_{\phi_R}(u,\delta)$ and $L_{\phi_f}(u,\delta)$.

3.1. Proximal Operators of Truncated Loss Functions

Definition 1 (Proximal Operator [35]). *Assume $f\colon \mathbb{R} \to \overline{\mathbb{R}}$ is a proper lower-semi-continuous loss function. The expression for the proximal operator of $f(u)$ at $x \in \mathbb{R}$ is defined as follows:*

$$prox_{\lambda f}(x) = \arg\min\left\{ f(u) + \frac{1}{2\lambda}\|u - x\|^2 \right\} \tag{16}$$

whenf$(u)$is a convex loss function, it presents a single-value proximal operator; whenf$(u)$is a non-convex loss function, it exhibits a multi-value proximal operator.

Lemma 1 ([36]). *When $f(u)$= $\log(1 + \frac{u}{\theta})$, let $v(x) = \arg\min\limits_{u\in\mathbb{R}}\left\{\log(1 + \frac{u}{\theta}) + \frac{1}{2\lambda}\|u - x\|^2\right\}$, $x \in \mathbb{R}$. The expressions for the proximal operators are as follows:*

$$v(x) = \begin{cases} \overline{v}(x) & (x-\theta)^2 - 4(\lambda - x\theta) \geq 0 \textit{ and } u > 0 \\ 0 & \textit{othersize} \end{cases} \tag{17}$$

$$\overline{v}(x) = \min\left\{0, \left[\frac{(x-\theta)+\sqrt{(x-\theta)^2-4(\lambda-x\theta)}}{2}\right]_+, \left[\frac{(x-\theta)-\sqrt{(x-\theta)^2-4(\lambda-x\theta)}}{2}\right]_+\right\}.$$

Lemma 2. *The explicit expression of the$L_{\phi_R}(u,\delta)$ proximal operator is as follows:*

1. *When $0 < \lambda < 2\delta v^2$, the explicit expression of the $L_{\phi_R}(u,\delta)$ proximal operator is presented as follows:*

$$prox_{\lambda L_R}(x) = \begin{cases} x & x > \delta v + \frac{\lambda}{2v} \\ \left\{x, x - \frac{\lambda}{v}\right\} & x = \delta v + \frac{\lambda}{2v} \\ x - \frac{\lambda}{v} & \frac{\lambda}{v} < x < \delta v + \frac{\lambda}{2v} \\ 0 & 0 \leq x \leq \frac{\lambda}{v} \\ x & x < 0 \end{cases} \tag{18}$$

2. *When $0 < \lambda < 2\delta v^2$, the explicit expression of the $L_{\phi_R}(u,\delta)$ proximal operator is as follows:*

$$prox_{\lambda L_R}(x) = \begin{cases} x & x > \sqrt{2\lambda\delta} \\ \{x, 0\} & x = \sqrt{2\lambda\delta} \\ 0 & 0 \leq x \leq \sqrt{2\lambda\delta} \\ x & x < 0 \end{cases} \tag{19}$$

Proof of Lemma 1. Equation (16) exhibits that $prox_{L_{\phi_R}}$ is a local minimum of the following piecewise function:

$$\varphi(u) = \begin{cases} \varphi_1(u) = \delta + \frac{1}{2\lambda}(u-x)^2 & u > \delta v \\ \varphi_2(u) = \delta + \frac{1}{2\lambda}(1-x)^2 & u = \delta v \\ \varphi_3(u) = \frac{u}{v} + \frac{1}{2\lambda}(u-x)^2 & 0 < u < \delta v \\ \varphi_4(u) = \frac{x^2}{2\lambda} & u = 0 \\ \varphi_5(u) = \frac{1}{2\lambda}(u-x)^2 & u < 0 \end{cases}$$

The minima of the piecewise functions $\varphi_1(u)$, $\varphi_2(u)$, $\varphi_3(u)$, $\varphi_4(u)$, and $\varphi_5(u)$ are located at $u_1^* = x$, $u_2^* = 1$, $u_3^* = x - \frac{\lambda}{v}$, $u_4^* = 0$, and $u_5^* = x$, with minimum values of $\varphi_1(u_1^*) = \delta$, $\varphi_2(u_2^*) = \delta + \frac{1}{2\lambda}(\delta v - x)^2$, $\varphi_3(u_3^*) = \frac{x}{v} - \frac{\lambda}{2v^2}$, $\varphi_4(u_4^*) = \frac{x^2}{2\lambda}$, and $\varphi_5(u_5^*) = 0$, respectively.

Since $0 < u_3^* = x - \frac{\lambda}{v} < v\delta$, it follows that $\frac{\lambda}{v} < x < v\delta + \frac{\lambda}{v}$. If $\varphi_1(u_1^*) < \varphi_3(u_3^*)$, then $x > \delta v + \frac{\lambda}{2v}$. When $\delta v + \frac{\lambda}{2v} \le \frac{\lambda}{v}$ and x is in the interval $\left(\frac{\lambda}{v}, v\delta + \frac{\lambda}{v}\right)$, $\varphi_1(u_1^*) < \varphi_3(u_3^*)$.

When $0 < \lambda < 2\delta v^2$, the following conclusion can be reached by comparing the values of $\varphi_1(u_1^*)$, $\varphi_2(u_2^*)$, $\varphi_3(u_3^*)$, $\varphi_4(u_4^*)$, and $\varphi_5(u_5^*)$.

(1.1) Since $x > v\delta + \frac{\lambda}{2v}$, we achieve $\min\{\varphi_2(u_2^*), \varphi_3(u_3^*), \varphi_4(u_4^*)\} > \varphi_1(u_1^*)$, which means $u^* = u_1^* = x$.

(1.2) Since $x = v\delta + \frac{\lambda}{2v}$, we obtain $\min\{\varphi_2(u_2^*), \varphi_4(u_4^*)\} > \varphi_1(u_1^*) = \varphi_3(u_3^*)$, which means $u^* = u_1^* = x$ or $u^* = u_3^* = x - \frac{\lambda}{v}$.

(1.3) Since $\frac{\lambda}{v} < x < v\delta + \frac{\lambda}{2v}$, we achieve $\min\{\varphi_1(u_1^*), \varphi_2(u_2^*), \varphi_4(u_4^*)\} > \varphi_3(u_3^*)$, which means $u^* = u_3^* = x - \frac{\lambda}{v}$.

(1.4) Since $0 \le x \le \frac{\lambda}{v}$, we obtain $\min\{\varphi_1(u_1^*), \varphi_2(u_3^*), \varphi_3(u_3^*)\} > \varphi_4(u_4^*)$, which means $u^* = u_4^* = 0$.

(1.5) Since $x < 0$, we achieve $\min\{\varphi_2(u_2^*), \varphi_4(u_4^*)\} > \varphi_5(u_5^*)$, which means $u^* = u_5^* = x$.

According to (1.1)–(1.5), Equation (18) can be derived.

When $\lambda \ge 2\delta v^2$, the following conclusion can be reached by comparing the values of $\varphi_1(u_1^*)$, $\varphi_2(u_2^*)$, $\varphi_3(u_3^*)$, $\varphi_4(u_4^*)$, and $\varphi_5(u_5^*)$.

(2.1) As $x > \sqrt{2\lambda\delta}$, we obtain $\min\{\varphi_2(u_2^*), \varphi_3(u_3^*), \varphi_4(u_4^*)\} > \varphi_1(u_1^*)$, which means $u^* = u_1^* = x$.

(2.2) As $x = \sqrt{2\lambda\delta}$, we obtain $\min\{\varphi_2(u_2^*), \varphi_3(u_3^*)\} > \varphi_1(u_1^*) = \varphi_4(u_4^*)$, which either means $u^* = u_1^* = x$ or $u^* = u_4^* = 0$.

(2.3) As $0 \le x < \sqrt{2\lambda\delta}$, we obtain $\min\{\varphi_1(u_1^*), \varphi_2(u_3^*), \varphi_3(u_3^*)\} > \varphi_4(u_4^*)$, which means $u^* = u_4^* = 0$.

(2.4) As $x < 0$, we obtain $\min\{\varphi_1(u_1^*), \varphi_2(u_2^*), \varphi_3(u_3^*), \varphi_4(u_4^*)\} > \varphi_5(u_5^*)$, which means $u^* = u_5^* = x$.

According to (2.1)–(2.4), Equation (19) can be derived. □

Lemma 3. *The expression representing the proximal operator of the truncation function is as follows:*

$$prox_{\lambda L}(x) = \begin{cases} x & x > \lambda' \textit{ and } \varphi_1(u_1^*) < \varphi_3(u_3^*) \textit{ and } \varphi_1(u_1^*) < \varphi_4(u_4^*) \\ \{x, v(x)\} & x = \lambda' \textit{ and } \varphi_1(u_1^*) = \varphi_3(u_3^*) < \varphi_4(u_4^*) \\ \{x, 0\} & x = \lambda' \textit{ and } \varphi_1(u_1^*) = \varphi_4(u_4^*) < \varphi_3(u_3^*) \\ v(x) & \omega_{down} < x < \omega_{up} \textit{ and } \varphi_1(u_1^*) > \varphi_3(u_3^*) \textit{ and } \varphi_3(u_3^*) < \varphi_4(u_4^*) \\ 0 & x \ge 0 \textit{ and } \varphi_3(u_3^*) > \varphi_4(u_4^*) \textit{ and } \varphi_1(u_1^*) > \varphi_4(u_4^*) \\ x & x < 0 \end{cases} \tag{20}$$

where u_1^*, u_2^*, u_3^*, and u_4^* represent the minimizers of the piecewise function, and $\varphi_1(u_1^*)$, $\varphi_2(u_2^*)$, $\varphi_3(u_3^*)$, and $\varphi_4(u_4^*)$, represent the minimal values of the piecewise function.

Proof of Lemma 3. It can be deduced from Equations (15) and (16) that $prox_{\lambda L}$ represents the local minimum of the following piecewise function:

$$\varphi(u) = \begin{cases} \varphi_1(u) = \delta + \frac{1}{2\lambda}(u-x)^2 & u > \lambda' \\ \varphi_2(u) = \delta + \frac{1}{2\lambda}(\lambda'-x)^2 & u = \lambda' \\ \varphi_3(u) = \phi(u) + \frac{1}{2\lambda}(u-x)^2 & 0 < u < \lambda' \\ \varphi_4(u) = \frac{x^2}{2\lambda} & u = 0 \\ \varphi_5(u) = \frac{1}{2\lambda}(u-x)^2 & x < 0 \end{cases}$$

Let $v(x) = \arg\min \phi(u) + \frac{1}{2\lambda}(u-x)^2$; the minimizers of the piecewise functions are $u_1^* = x$, $u_2^* = \lambda'$, $u_3^* = v(x)$, $u_4^* = 0$, and $u_5^* = x$, and their minimal values are $\varphi_1(u_1^*) = \delta$, $\varphi_2(u_2^*) = \delta + \frac{1}{2\lambda}(\lambda'-x)^2$, $\varphi_3(u_3^*) = \varphi_3(v(x))$, $\varphi_4(u_4^*) = \frac{x^2}{2\lambda}$, and $\varphi_5(u_5^*) = 0$, respectively. From $0 < u_3^* = v(x) < \lambda'$, it follows that $\omega_{down} < x < \omega_{up}$.

When $x \geq 0$, the stage function's minimal values are $\varphi_1(u_1^*)$,$\varphi_2(u_2^*)$,$\varphi_3(u_3^*)$, and $\varphi_4(u_4^*)$; when $x < 0$, the stage function's minimal values are $\varphi_2(u_2^*)$, $\varphi_4(u_4^*)$, and $\varphi_5(u_5^*)$. Thus, we can observe that $\varphi_2(u_2^*) \geq \varphi_1(u_1^*)$. If $\varphi_1(u_1^*) \geq \varphi_3(u_3^*)$, then $\varphi_2(u_2^*) \geq \varphi_1(u_1^*) \geq \varphi_3(u_3^*)$; similarly, if $\varphi_1(u_1^*) \geq \varphi_4(u_4^*)$, then $\varphi_2(u_2^*) \geq \varphi_1(u_1^*) \geq \varphi_4(u_4^*)$. We can determine the following conclusions by comparing the values of $\varphi_1(u_1^*)$, $\varphi_2(u_2^*)$, $\varphi_3(u_3^*)$, $\varphi_4(u_4^*)$, and $\varphi_5(u_5^*)$:

(1.1) When the conditions of $x > \lambda'$, $\varphi_1(u_1^*) < \varphi_3(u_3^*)$, and $\varphi_1(u_1^*) < \varphi_4(u_4^*)$ are met, and it follows that $u^* = u_1^* = x$;
(1.2) When the condition of $x = \lambda'$ is met, if $\varphi_1(u_1^*) = \varphi_3(u_3^*) < \varphi_4(u_4^*)$ is true, then $u^* = u_1^* = x$ or $u^* = u_3^* = v(x)$ can be derived;
(1.3) When the condition of $x = \lambda'$ is met, if $\varphi_1(u_1^*) = \varphi_4(u_4^*) < \varphi_3(u_3^*)$ is true, then $u^* = u_1^* = x$ or $u^* = u_4^* = 0$ can be derived;
(1.4) When the conditions of $\omega_{down} < x < \omega_{up}$, $\varphi_1(u_1^*) > \varphi_3(u_3^*)$, and $\varphi_3(u_3^*) < \varphi_4(u_4^*)$ are met, it follows that $u^* = u_3^* = v(x)$;
(1.5) When the conditions of $x \geq 0$, $\varphi_1(u_1^*) > \varphi_4(u_4^*)$, and $\varphi_4(u_4^*) < \varphi_3(u_3^*)$ are met, it follows that $u^* = u_4^* = 0$;
(1.6) When the condition of $x < 0$ is met, it follows that $u^* = u_5^* = x$.

Thus, it is possible to successfully derive Equation (20). □

3.2. *The Use of the Proximal Operator Algorithm to Solve Truncated Loss Functions*

When $\phi(u)$ in the truncated loss function is a monotonic and non-piecewise function, and $v(x) = \arg\min \phi(u) + \frac{1}{2\lambda}(u-x)^2$ can be expressed explicitly, the proximal operator of the truncated loss function can be calculated using Formula (20). In practical applications, however, it is sometimes impossible to obtain the explicit expression for $v(x)$; for example, $\phi_L(u) = \exp(ayu) - ayu - 1$ does not provide an explicit expression. The calculation of the proximal operator in such scenarios is discussed below.

For $\arg\min \phi(u) + \frac{1}{2\lambda}(u-x)^2$, if it is smooth and has a second derivative, the problem is a smooth unconstrained optimization problem. Newton's method is used to solve for the minimum of $\varphi(u)$ in unconstrained optimization problems due to its high convergence rate. The gradient and Hessian matrix for problem (16) can be expressed as follows:

$$g = \frac{\partial \phi(u)}{\partial u} + \frac{1}{\lambda}(u-x) \tag{21}$$

$$H = \frac{\partial^2 \phi(u)}{\partial u^2} + \frac{1}{\lambda} I \tag{22}$$

The minimizer u_3^* and the minimal value $\varphi_3(u_3^*)$ can be obtained with Newton's method for x.

If an explicit expression for $v(x)$ cannot be achieved, the calculation of the proximal operator follows the same process as Lemma 3. Once the minimizers u_3^* and $\varphi_3(u_3^*)$ are obtained, the proximal operator can be calculated. When the conditions of $0 < u_3^* < \lambda'$, $\varphi_1(u_1^*) > \varphi_3(u_3^*)$, and $\varphi_3(u_3^*) < \varphi_4(u_4^*)$ are met, we can derive $u^* = u_3^*$. Therefore, Formula (20) can be modified to express the following:

$$prox_{\lambda L}(x) = \begin{cases} x & x > \lambda' \text{ and } \varphi_1(u_1^*) < \varphi_3(u_3^*) \text{ and } \varphi_1(u_1^*) < \varphi_4(u_4^*) \\ \{x, u_3^*\} & x = \lambda' \text{ and } \varphi_1(u_1^*) = \varphi_3(u_3^*) < \varphi_4(u_4^*) \\ \{x, 0\} & x = \lambda' \text{ and } \varphi_1(u_1^*) = \varphi_4(u_4^*) < \varphi_3(u_3^*) \\ u_3^* & 0 < u_3^* < \lambda' \text{ and } \varphi_1(u_1^*) > \varphi_3(u_3^*) \text{ and } \varphi_3(u_3^*) < \varphi_4(u_4^*) \\ 0 & x \geq 0 \text{ and } \varphi_3(u_3^*) > \varphi_4(u_4^*) \text{ and } \varphi_1(u_1^*) > \varphi_4(u_4^*) \\ x & x < 0 \end{cases} \quad (23)$$

Based on the analysis presented above, the algorithm for solving the proximal operator of the truncated loss function is as following Algorithm 1:

Algorithm 1: Algorithm for solving the proximal operator of the truncated loss function

```
Input : $x, \delta, \lambda, \phi(u)$
Output : Proximal operator $u^*$
1 : Choose an initial point $u^{(0)} = x$.
2 : While $t \leq MAX$ do
3 : Calculate $g^{(t)}$ according to Formula (21).
4 : If $\|g^{(t)}\| < eps$, then stop the loop, the approximate solution $u_3^* = u^{(t+1)}$ is obtained.
5 :     Calculate $H^{(t)}$ according to Formula (22).
6 :     Calculate $\overline{d}$, the Newton iteration direction, according to the Newton iteration
        equation $H^{(t)}d = -g^{(t)}$.
7 :     Solve for the step size $\theta$ using the backtracking Armijo method, and update
        $u^{(t)} = u^{(t-1)} + \theta\overline{d}$.
8 :     Set $t = t + 1$.
9 : End
10 : Calculate $\varphi(u_3^*)$ according to $u_3^*$.
11 : Calculate the proximal operator $u^*$ according to Formula (23).
```

4. Robust SVDD Model

Formula (1), for the SVDD formula, can be rewritten as follows:

$$\min_{R,\mu,\xi_i} R^2 + C\sum_i [u_i]_+, u_i = \|\varphi(x_i) - \mu\|^2 - R^2 \quad (24)$$

where $[u]_+ = \max(0, u)$ represents the hinge loss function. Since the hinge loss function is sensitive to outliers, it can be replaced with the truncated loss function from Formula (15). Thus, the objective function for obtaining the robust SVDD model is as follows:

$$\min_{R,\mu,\xi_i} R^2 + C\sum_i (\phi(u) - (\delta - \phi(u))_+), u_i = \|\varphi(x_i) - \mu\|^2 - R^2 \quad (25)$$

As the truncated loss function is non-differentiable, solving the objective function of the robust SVDD model is a non-convex optimization problem, and it cannot be solved using standard SVDD model methods.

Theorem 1 (Nonparametric Representation Theorem [37])**.** *Suppose we are designated a non-empty set χ; a positive definite real-valued kernel K: $\chi \times \chi \to \mathbb{R}$; a training sample $(x_i, y_i)(i \in \mathbb{N}_m) \in \chi \times \mathbb{R}$; a strictly monotonically increasing real-valued function g on $[0, +\infty]$; an arbitrary cost function c: $(\chi \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$; and a class of functions:*

$$F = \left\{ f \in \mathbb{R}^{\chi} \middle| f(\cdot) = \sum_{i=1}^{m} \beta_i K(z_i, \cdot), \beta_i \in \mathbb{R}, z_i \in \chi, \|f\| \leq \infty \right\}$$

In this scenario, $\| \cdot \|$ represents the norm in RKHS, H_k associated with $K(\cdot, \cdot)$, i.e., for any $z_i \in \chi$.

$$\|\sum_{i=1}^{\infty} \beta_i K(z_i, \cdot)\|^2 = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \beta_i \beta_j K(z_i, z_j) \tag{26}$$

Then, any $f \in F$ minimizing the regularized risk function

$$c((x_1, x_1, y_1, f(x_1)), \cdots, (x_m, y_m, f(x_m))) + g(\|f\|) \tag{27}$$

admits a representation of $f(\cdot) = -\sum_{i=1}^{m} a_i y_i K(x_i, \cdot)$, where $a_i \in \mathbb{R}(i \in \mathbb{N}_m)$ represents coefficients of f in RKHS H_k.

A set of vectors a exists in the nonparametric representation theorem, where the center $\mu = \sum_{i=1}^{n} a_i \phi(x_i)$ is the optimal solution for problem (25). Therefore, Formula (25) can be transformed into the following:

$$\begin{aligned} &\min_{R,u_i} R^2 + C\sum_i L(u_i) \\ &u_i = K(x_i, x_i) - 2\sum_{j=1}^{n} a_j K(x_i, x_j) + \sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k K(x_j, x_k) - R^2 \end{aligned} \tag{28}$$

Formula (28) represents the single-class SVDD model. To obtain data that include negative samples, these samples must be integrated into the SVDD model; then, the center is $\mu = \sum_{i=1}^{n} a_i y_i \phi(x_i)$, and the objective function of the robust SVDD model is as follows:

$$\begin{aligned} &\min_{R,\mu,\xi_i} R^2 + \sum_i C_{y_i} L(u_i) \\ &u_i = y_i K(x_i, x_i) - 2y_i \sum_{j=1}^{n} a_j y_j K(x_i, x_j) + y_i \sum_{j=1}^{n}\sum_{k=1}^{n} a_j y_j a_k y_k K(x_j, x_k) - y_i R^2 \end{aligned} \tag{29}$$

when $a_y = a. * \mathrm{y}$, it follows that $\mu = \sum_{i=1}^{n} a_{y_i} \phi(x_i)$. Problem (29) is rewritten as the following matrix form:

$$\begin{aligned} &\min_{R,u} R^2 + CL_{\phi}(u) \\ &u = K_a - 2K_b a_y + a_y^T K a_y D - R^2 D, R^2 > 0 \end{aligned} \tag{30}$$

where $K_a = diag(K). * y$, $K_b = [y_1 * K(1, \cdot), y_2 * K(2, \cdot), \cdots, y_n * K(n, \cdot)]$, $D = [y_1, y_2 \cdots, y_n]^T$, and $C = \left[C_{y_1}, C_{y_2} \cdots, C_{y_n}\right]$. This study discusses the use of the SVDD model as a solution for addressing data with negative samples, and the Lagrangian function for problem (30) is as follows:

$$L_{\beta}\left(R^2, a_y, u, \eta, \beta\right) = R^2 + CL_{\phi}(u) + \left\langle \eta, K_a - 2K_b a_y + a_y^T K a_y D - R^2 D - u \right\rangle \tag{31}$$

The KKT conditions for problem (30) are provided below:

$$\begin{cases} K_a - 2K_b a_y^* + {a_y^*}^T K a_y^* D - R^{2^*} D - u^* = 0 \\ 0 = 1 - {\eta^*}^T D \\ 0 = \left(2D{a_y^*}^T K - 2{K_b}^T\right){\eta^*}^T \\ 0 \in \partial CL_{\phi}(u^*) - \eta^* \end{cases} \tag{32}$$

where $\left(R^{2^*}, a_y^*, u^*\right)$ represents any KKT point.

The generalized non-smooth optimization problem can be represented by the following Formula [38]:

$$\begin{aligned} &\min_{R,u} f(s) + cl(q) \\ &q + g(s) = 0 \end{aligned} \tag{33}$$

where $f(\cdot)$ and $g(\cdot)$ are continuously differentiable functions on $\mathbb{R}^n \to \mathbb{R}$, and $l(q)$ represents a non-smooth function on $\mathbb{R}^n \to \mathbb{R}$. Problem (31) is considered as a form of the aforementioned generalized non-smooth optimization problem, where $s = \left[R^2, a_y\right]$, $q = u$, $f(s) = s_1$, and $g(s) = -K_a + 2K_b s_2 - s_2^T K s_2 D + s_1 D$, representing the optimization model's constraints, are nonlinear equality constraints. In this study, the fast ADMM algorithm was employed to solve problem (33), and the algorithm will be introduced in a subsequent chapter.

5. Fast ADMM Algorithm

The previous section presented the optimization mathematical model of the robust SVDD model. Since the optimization mathematical model includes nonlinear equality constraints, the fast ADMM method was used to solve Formula (33). The augmented Lagrangian function of the robust SVDD model is as follows:

$$L_\beta\left(R^2, a_y, u, \eta, \beta\right) = R^2 + CL_\phi(u) + \left\langle \eta, g\left(a_y\right) - R^2 D - u \right\rangle + \frac{\beta}{2}\left\|g\left(a_y\right) - R^2 D - u\right\|^2 \tag{34}$$

where $\lambda \in \mathbb{R}^m$ represents the vector of Lagrange multipliers, and $\beta > 0$ represents the penalty factor, $g\left(a_y\right) = K_a - 2K_b a_y + a_y^T K a_y D$.

5.1. Fast ADMM Algorithm Framework

Based on the augmented Lagrangian function previously presented, we obtained the following framework for the fast ADMM algorithm:

$$u^{k+1} = \arg\min_{u \in \mathbb{R}^m} L_\beta\left(R^{2^k}, a_y^k, u, \eta^k\right) \tag{35}$$

$$a_y^{k+1} = \arg\min_{a_y \in \mathbb{R}^m} L_\beta\left(R^{2^k}, a_y, u^{k+1}, \eta^k\right) + \Delta\varphi_1\left(a_y, a_y^k\right) \tag{36}$$

$$R^{2k+1} = \arg\min_{R^2 \in \mathbb{R}^m} L_\beta\left(R^2, a_y^{k+1}, u^{k+1}, \eta^k\right) + \Delta\varphi_2\left(R^2, R^{2^k}\right) \tag{37}$$

$$\eta^{k+1} = \eta^k + \beta\left(g\left(a_y^{k+1}\right) - R^{2^{k+1}} D - u^{k+1}\right) \tag{38}$$

1. **Computing u^{k+1}**

The solution of u^{k+1} in Formula (35) is equivalent to the following problem:

$$\begin{aligned} u^{k+1} &= \arg\min_{u \in \mathbb{R}^m} CL_\phi(u) + \tfrac{\beta}{2}\left\|\left(g\left(a_y^k\right) - R^{2^k} D + \tfrac{\eta^k}{\beta}\right) - u\right\|^2 \\ &= \arg\min_{u \in \mathbb{R}^m} CL_\phi(u) + \tfrac{\beta}{2}\left\|z^k - u\right\|^2 \\ &= prox_{\frac{c}{\beta} L_\phi}\left(z^k\right) \end{aligned} \tag{39}$$

where $z^k = g\left(a_y^k\right) - R^{2^k} D + \frac{\eta^k}{\beta}$. Therefore, the calculation of u^{k+1} can be transformed into the computation of the proximal operator, which can be determined using Formula (20) or Algorithm 1.

2. **Computing a_y^{k+1}**

When solving for variable a_y^{k+1}, if a closed-form solution for this subproblem is not obtained, an optimization algorithm must be used for the iterative solution, which results in a slow computational speed. Thus, to solve this problem in a more efficient manner, we used a linearization technique.

Given the convex differentiable function φ, the Bregman distance between x and y,$x, y \in dom(\varphi)$ is defined as follows:

$$\Delta\varphi(x,y) = \varphi(x) - \varphi(y) - (\nabla\varphi(y), x - y) \tag{40}$$

When $\varphi(x) = \frac{\mu}{2}\|x\|^2 - \frac{\beta}{2}\|g(x) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\|^2$, the Bregman distance $\Delta\varphi(x,y)$ is as follows:

$$\begin{aligned}\Delta\varphi(x,y) = \frac{\mu}{2}\|x-y\|^2 - \frac{\beta}{2}\|g(x) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\|^2 + \frac{\beta}{2}\|g(y) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\|^2 \\ + \left\langle \beta\nabla g(y)\left(g(y) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right), x - y\right\rangle\end{aligned} \tag{41}$$

Formula (36) is equivalent to the following:

$$\begin{aligned}a_y^{k+1} = \arg\min_{a_y \in \mathbb{R}^m} \frac{\mu}{2}\|a_y - a_y^k\|^2 + \frac{\beta}{2}\|g\left(a_y^k\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\|^2 \\ + \left\langle \beta\nabla g\left(a_y^k\right)\left(g\left(a_y^k\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right), a_y - a_y^k\right\rangle\end{aligned} \tag{42}$$

With the use of Formula (42), we obtain

$$a_y^{k+1} = a_y^k - \frac{\beta\nabla g\left(a_y^k\right)}{\mu}\left(g\left(a_y^k\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right) \tag{43}$$

where $\mu \geq \beta\|\nabla g\left(a_y^k\right)\|^2$.

3. **Computing R^{2k+1}**

If $\varphi(x) = \frac{1}{2}\|x\|^2$, based on Formula (40), we obtain

$$\Delta\varphi_2\left(R^2, R^{2^k}\right) = \frac{1}{2}\|R^2 - R^{2^k}\| \tag{44}$$

Formula (37) is equivalent to

$$R^{2^{k+1}} = \arg\min_{R^2 \in \mathbb{R}^m} R^2 + \frac{\beta}{2}\|g\left(a_y^{k+1}\right) - R^2D - u^{k+1} + \frac{\eta^k}{\beta}\|^2 + \frac{1}{2}\|R^2 - R^{2^k}\|^2 \tag{45}$$

Formula (45) represents a convex quadratic programming problem. By solving the following Equation (46), we also solve Formula (45):

$$0 = 1 - \beta D^T\left(g\left(a_y^{k+1}\right) - R^2D - u^{k+1} + \frac{\eta^k}{\beta}\right) + R^2 - R^{2^k} \tag{46}$$

The value of $R^{2^{k+1}}$ is directly updated using the following formula:

$$R^{2^{k+1}} = \frac{\left(\beta D^T\left(g\left(a_y^{k+1}\right) - u^{k+1} + \frac{\eta^k}{\beta}\right) - 1 + R^{2^k}\right)}{(1 + \beta D^TD)} \tag{47}$$

4. Computing η^{k+1}

η^{k+1} can be calculated using Formula (38). When $\eta^{k+1} = 0$, the Lagrange multiplier is removed.

Base on above analysis, the framework of our method can be summarized in Algorithm 2.

Algorithm 2: Fast ADMM Algorithm

```
Input : $K_a,\ K_b,\ D$
Output : $\left(R^{2^k}, a_y^k\right)$
1 : Take an initial point $\left(R^{2^0}, a_y^0, u^0, \eta^0\right)$, and $k_{MAX}$.
2 : While the stop condition is not satisfied and $k \le k_{MAX}$ perform the following
3 :     Calculate $a_y^k$ according to Formula (43).
4 :     Calculate $R^{2^k}$ according to Formula (47).
5 :     If the proximal operator of $\phi(u)$ has an explicit expression, use Formula (20) for the
        calculation $u^k$
6 :     If the proximal operator of $\phi(u)$ does not have an explicit expression, use
        Algorithm 1 for the calculation $u^k$
7 :     Calculate $\eta^k$ according to Formula (38).
8 :     Update $k = k + 1$.
9: End
10: Output the final solution $\left(R^{2^k}, a_y^k\right)$.
```

5.2. Global Convergence Analysis of the Fast ADMM Algorithm

In this section, we provide a convergence analysis of the fast ADMM algorithm. Specifically, the convergence of the algorithm is discussed using Lemma 7, and Lemma 7 is proven. According to Formulas (35)–(38), it can be observed that the optimality conditions for each update iteration of the fast ADMM algorithm can be written as follows:

$$\begin{cases} 0 \in \partial CL_\phi\left(u^{k+1}\right) - \eta^k - \beta\left(g\left({a_y}^k\right) - R^{2^k}D - u^{k+1}\right) \\ 0 = \mu\left(a_y^{k+1} - a_y^k\right) - \beta\nabla g\left(a_y^k\right)\left(g\left(a_y^k\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right) \\ 0 = 1 - D^T\eta^k - \beta D^T\left(g\left({a_y}^{k+1}\right) - R^{2^{k+1}}D - u^{k+1}\right) + R^{2^{k+1}} - R^{2^k} \\ \eta^{k+1} = \eta^k + \beta\left(g\left({a_y}^{k+1}\right) - R^{2^{k+1}}D - u^{k+1}\right) \end{cases} \tag{48}$$

where $\nabla g(a_y) = 2D{a_y}^T K - 2{K_b}^T$.

Lemma 4. *Assume $w^k = \left(R^{2^k}, a_y^k, u^k, \eta^k\right)$ is a sequence generated using the fast ADMM algorithm; then, for any $k \ge 1$, we obtain the following:*

$$\begin{aligned} L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^k\right) \le L_\beta\left(R^{2^k}, a_y^{k+1}, u^{k+1}, \eta^k\right) - \tfrac{\beta}{2}\|R^{2^{k+1}}D - R^{2^k}D\|^2 \\ - \tfrac{1}{2}\|R^{2^{k+1}} - R^{2^k}\|^2 - \tfrac{1}{2}\|R^{2^k} - R^{2^{k-1}}\|^2 \end{aligned} \tag{49}$$

where $\lambda_{\min}(DD^T)$ represents the strictly positive minimum eigenvalue of DD^T.

Proof of Lemma 4. According to the optimality conditions of the iteration, the following equation is relevant:

$$D^T\eta^{k+1} = 1 + R^{2^{k+1}} - R^{2^k}$$

According to the Cauchy–Schwarz inequality, it follows that

$$\begin{aligned} \lambda_{\min}(DD^T)\|\eta^{k+1} - \eta^k\|^2 \le \|D^T\eta^{k+1} - D^T\eta^k\|^2 = \|R^{2^{k+1}} - R^{2^k} - \left(R^{2^k} - R^{2^{k-1}}\right)\|^2 \\ \le \|R^{2^{k+1}} - R^{2^k}\|^2 + \|R^{2^k} - R^{2^{k-1}}\|^2 \end{aligned}$$

Thus,

$$\frac{1}{\beta}\|\eta^{k+1}-\eta^{k}\|^{2} \le \frac{1}{\lambda_{\min}(DD^{T})\beta}\|R^{2^{k+1}}-R^{2^{k}}\|^{2}+\frac{1}{\lambda_{\min}(DD^{T})\beta}\|R^{2^{k}}-R^{2^{k-1}}\|^{2}.$$

□

Lemma 5. *Assume $w^{k}=\left(R^{2^{k}},a_{y}^{k},u^{k},\eta^{k}\right)$ is a sequence generated using the fast ADMM algorithm; then, for any $k\ge 1$, the following is true:*

$$\begin{aligned}L_{\beta}\left(R^{2^{k+1}},a_{y}^{k+1},u^{k+1},\eta^{k}\right) \le{}& L_{\beta}\left(R^{2^{k}},a_{y}^{k+1},u^{k+1},\eta^{k}\right)-\tfrac{\beta}{2}\|R^{2^{k+1}}D-R^{2^{k}}D\|^{2}\\ &-\tfrac{1}{2}\|R^{2^{k+1}}-R^{2^{k}}\|^{2}-\tfrac{1}{2}\|R^{2^{k}}-R^{2^{k-1}}\|^{2}\end{aligned}\tag{50}$$

Proof of Lemma 5. From the definition of the augmented Lagrangian function presented in Formula (34), it can be argued that

$$\begin{aligned}&L_{\beta}\left(R^{2^{k+1}},a_{y}^{k+1},u^{k+1},\eta^{k}\right)-L_{\beta}\left(R^{2^{k}},a_{y}^{k+1},u^{k+1},\eta^{k}\right)\\ &=R^{2^{k+1}}-R^{2^{k}}+\left\langle\eta^{k},\left(R^{2^{k}}-R^{2^{k+1}}\right)D\right\rangle+\tfrac{\beta}{2}\|g\left(a_{y}^{k+1}\right)-R^{2^{k+1}}D-u^{k+1}\|^{2}-\tfrac{\beta}{2}\|g\left(a_{y}^{k+1}\right)-R^{2^{k}}D-u^{k+1}\|^{2}\\ &+\tfrac{1}{2}\|R^{2^{k+1}}-R^{2^{k}}\|^{2}-\tfrac{1}{2}\|R^{2^{k}}-R^{2^{k-1}}\|^{2}\end{aligned}\tag{51}$$

Formula (48) shows the following:

$$\mathrm{g}\left(a_{y}^{k+1}\right)-R^{2^{k}}D-u^{k+1}=\frac{1}{\beta}\left(\eta^{k+1}-\eta^{k}\right)+\left(R^{2^{k+1}}D-R^{2^{k}}D\right)\tag{52}$$

Therefore,

$$\begin{aligned}&\tfrac{\beta}{2}\|\varphi\left(a_{y}^{k+1}\right)-R^{2^{k}}D-u^{k+1}\|^{2}=\tfrac{\beta}{2}\|\tfrac{1}{\beta}\left(\eta^{k+1}-\eta^{k}\right)+\left(R^{2^{k+1}}D-R^{2^{k}}D\right)\|^{2}\\ &=\tfrac{1}{2\beta}\|\eta^{k+1}-\eta^{k}\|^{2}+\tfrac{\beta}{2}\|R^{2^{k+1}}D-R^{2^{k}}D\|^{2}+\left\langle\eta^{k+1}-\eta^{k},R^{2^{k+1}}D-R^{2^{k}}D\right\rangle\end{aligned}\tag{53}$$

By substituting Formula (53) into (51), the following is achieved:

$$\begin{aligned}&L_{\beta}\left(R^{2^{k+1}},a_{y}^{k+1},u^{k+1},\eta^{k}\right)-L_{\beta}\left(R^{2^{k}},a_{y}^{k+1},u^{k+1},\eta^{k}\right)\\ &=R^{2^{k+1}}-R^{2^{k}}+\left\langle\eta^{k},\left(R^{2^{k}}-R^{2^{k+1}}\right)D\right\rangle-\tfrac{\beta}{2}\|R^{2^{k+1}}D-R^{2^{k}}D\|^{2}-\left\langle\eta^{k+1}-\eta^{k},R^{2^{k+1}}D-R^{2^{k}}D\right\rangle\\ &+\tfrac{1}{2}\|R^{2^{k+1}}-R^{2^{k}}\|^{2}-\tfrac{1}{2}\|R^{2^{k}}-R^{2^{k-1}}\|^{2}\\ &=-\tfrac{\beta}{2}\|R^{2^{k+1}}D-R^{2^{k}}D\|^{2}+\left\langle 1-D^{T}\eta^{k+1},R^{2^{k+1}}-R^{2^{k}}\right\rangle+\tfrac{1}{2}\|R^{2^{k+1}}-R^{2^{k}}\|^{2}-\tfrac{1}{2}\|R^{2^{k}}-R^{2^{k-1}}\|^{2}\\ &=-\tfrac{\beta}{2}\|R^{2^{k+1}}D-R^{2^{k}}D\|^{2}-\tfrac{1}{2}\|R^{2^{k+1}}-R^{2^{k}}\|^{2}-\tfrac{1}{2}\|R^{2^{k}}-R^{2^{k-1}}\|^{2}\end{aligned}$$

Thus, the proof is completed. □

Lemma 6. *Assume $w^{k}=\left(R^{2^{k}},a_{y}^{k},u^{k},\eta^{k}\right)$ is a sequence generated using the fast ADMM algorithm; then, for $k\ge 1$, we obtain the following:*

$$\begin{aligned}L_{\beta}\left(R^{2^{k}},a_{y}^{k+1},u^{k+1},\eta^{k}\right) \le{}& L_{\beta}\left(R^{2^{k}},a_{y}^{k},u^{k+1},\eta^{k}\right)+\tfrac{\beta\|\nabla g\left(a_{y}^{k}\right)\|^{2}}{2}\|a_{y}^{k}-a_{y}^{k-1}\|^{2}\\ &-\tfrac{\mu}{2}\|a_{y}^{k+1}-a_{y}^{k}\|^{2}-\tfrac{\mu}{2}\|a_{y}^{k}-a_{y}^{k-1}\|^{2}\end{aligned}\tag{54}$$

Proof of Lemma 6. The definition of the augmented Lagrangian function in Formula (34) shows the following:

$$\begin{aligned}
&L_\beta\left(R^{2^k}, a_y^{k+1}, u^{k+1}, \eta^k\right) - L_\beta\left(R^{2^k}, a_y^k, u^{k+1}, \eta^k\right) \\
&= \frac{\beta}{2}\|g\left(a_y^k\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\|^2 - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\mu}{2}\|a_y^k - a_y^{k-1}\|^2 - \frac{\beta}{2}\|g\left(a_y^{k-1}\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\|^2 \\
&-\left\langle \beta\nabla g\left(a_y^{k-1}\right)\left(g\left(a_y^{k-1}\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right), a_y^k - a_y^{k-1}\right\rangle \\
&= \frac{\beta}{2}\|g\left(a_y^k\right) - g\left(a_y^{k-1}\right)\|^2 - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\mu}{2}\|a_y^k - a_y^{k-1}\|^2 + \beta\left\langle g\left(a_y^{k-1}\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}, g\left(a_y^k\right) - g\left(a_y^{k-1}\right)\right\rangle \\
&-\left\langle \beta\nabla g\left(a_y^{k-1}\right)\left(g\left(a_y^{k-1}\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right), a_y^k - a_y^{k-1}\right\rangle
\end{aligned} \tag{55}$$

According to the first-order condition for convex functions, and since $g\left(a_y\right)$ is a convex function, we obtain the following:

$$g\left(a_y^{k+1}\right) - g\left(a_y^k\right) \le \left\langle \partial g\left(a_y^{k+1}\right), a_y^{k+1} - a_y^k\right\rangle \tag{56}$$

The substitution of Formula (56) yields the following:

$$\begin{aligned}
&L_\beta\left(R^{2^k}, a_y^{k+1}, u^{k+1}, \eta^k\right) - L_\beta\left(R^{2^k}, a_y^k, u^{k+1}, \eta^k\right) \\
&= \frac{\beta}{2}\|g\left(a_y^k\right) - g\left(a_y^{k-1}\right)\|^2 - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\mu}{2}\|a_y^k - a_y^{k-1}\|^2 + \beta\left\langle g\left(a_y^{k-1}\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}, g\left(a_y^k\right) - g\left(a_y^{k-1}\right)\right\rangle \\
&-\left\langle \beta\nabla g\left(a_y^{k-1}\right)\left(g\left(a_y^{k-1}\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right), a_y^k - a_y^{k-1}\right\rangle \\
&\le \frac{\beta}{2}\|g\left(a_y^k\right) - g\left(a_y^{k-1}\right)\|^2 - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\mu}{2}\|a_y^k - a_y^{k-1}\|^2 \\
&\le \frac{\beta\|\nabla g\left(a_y^k\right)\|^2}{2}\|a_y^k - a_y^{k-1}\|^2 - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\mu}{2}\|a_y^k - a_y^{k-1}\|^2
\end{aligned}$$

Thus, the proof is complete. □

Lemma 7. *Assume $w^k = \left(R^{2^k}, a_y^k, u^k, \eta^k\right)$ is a sequence generated using the fast ADMM algorithm; if $\mu \ge \beta\|\nabla g\left(a_y^k\right)\|^2$ and $\frac{1}{\lambda_{\min}\left(DD^T\right)\beta} \le \frac{1}{2}$, then for any $k \ge 1$, we obtain the following:*

$$L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^{k+1}\right) \le L_\beta\left(R^{2^k}, a_y^k, u^k, \eta^k\right) - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\beta}{2}\|R^{2^{k+1}}D - R^{2^k}D\|^2 \tag{57}$$

Proof of Lemma 7. The definition of the augmented Lagrangian function in Formula (36) shows the following:

$$\begin{aligned}
L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^{k+1}\right) &= L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^k\right) + \left\langle \eta^{k+1} - \eta^k, \varphi\left(a_y^{k+1}\right) - R^{2^{k+1}}D - u^{k+1}\right\rangle \\
&= L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^k\right) + \frac{1}{\beta}\|\eta^{k+1} - \eta^k\|^2
\end{aligned}$$

Since $L_\beta\left(R^{2^k}, a_y^k, u, \eta^k\right)$ is a global minimum with respect to the variable u^{k+1}, then

$$L_\beta\left(R^{2^k}, a_y^k, u^{k+1}, \eta^k\right) \le L_\beta\left(R^{2^k}, a_y^k, u^k, \eta^k\right)$$

The combination of Lemmas 5 and 6 produces the following:

$$\begin{aligned}
&L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^{k+1}\right) = L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^k\right) + \frac{1}{\beta}\|\eta^{k+1} - \eta^k\|^2 \\
&= L_\beta\left(R^{2^k}, a_y^{k+1}, u^{k+1}, \eta^k\right) - \frac{\beta}{2}\|R^{2^{k+1}}D - R^{2^k}D\|^2 + \frac{1}{\beta}\|\eta^{k+1} - \eta^k\|^2 - \frac{1}{2}\|R^{2^{k+1}} - R^{2^k}\|^2 - \frac{1}{2}\|R^{2^k} - R^{2^{k-1}}\|^2 \\
&\le\ L_\beta\left(R^{2^k}, a_y^k, u^k, \eta^k\right) - \frac{\beta}{2}\|R^{2^{k+1}}D - R^{2^k}D\|^2 - \frac{1}{2}\|R^{2^{k+1}} - R^{2^k}\|^2 - \frac{1}{2}\|R^{2^k} - R^{2^{k-1}}\|^2 + \frac{1}{\beta}\|\eta^{k+1} - \eta^k\|^2 \\
&+\frac{\beta\|\nabla g\left(a_y^k\right)\|^2}{2}\|a_y^k - a_y^{k-1}\|^2 - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\mu}{2}\|a_y^k - a_y^{k-1}\|
\end{aligned}$$

When $\mu \geq \beta \|\nabla g\left(a_y^k\right)\|^2$ and $\frac{1}{\lambda_{\min}\left(DD^T\right)\beta} \leq \frac{1}{2}$, it follows that

$$L_\beta\left(R^{2^{k+1}}, a_y^{k+1}, u^{k+1}, \eta^{k+1}\right) \leq L_\beta\left(R^{2^k}, a_y^k, u^k, \eta^k\right) - \frac{\mu}{2}\|a_y^{k+1} - a_y^k\|^2 - \frac{\beta}{2}\|R^{2^{k+1}}D - R^{2^k}D\|^2$$

When the conditions of $\mu \geq \beta \|\nabla g\left(a_y^k\right)\|^2$ and $\frac{1}{\lambda_{\min}\left(DD^T\right)\beta} \leq \frac{1}{2}$ are met, according to Lemma 7, $L_\beta\left(R^2, a_y, u, \eta\right)$ monotonically decreases, thus causing the fast ADMM algorithm to converge. □

5.3. Fast ADMM Algorithm Termination Conditions

Based on Formula (35), it can be observed that the u^{k+1} that minimizes $L_\beta\left(R^{2^k}, a_y^k, u, \eta^k\right)$ can be derived as follows:

$$\begin{aligned} 0 &\in \partial CL_\phi\left(u^{k+1}\right) - \beta\left(g\left(a_y^k\right) - R^{2^k}D - u^{k+1} + \frac{\eta^k}{\beta}\right) \\ &= \partial CL_\phi\left(u^{k+1}\right) - \eta^{k+1} - \beta\left(g\left(a_y^{k+1}\right) - g\left(a_y^k\right)\right) - \beta\left(R^{2^{k+1}}D - R^{2^k}D\right) \end{aligned} \tag{58}$$

where

$$\beta\left(g\left(a_y^{k+1}\right) - g\left(a_y^k\right)\right) + \beta\left(R^{2^{k+1}}D - R^{2^k}D\right) \in \partial CL_\phi\left(u^{k+1}\right) - \eta^{k+1} \tag{59}$$

Thus, we obtain the following equation:

$$S_1^{k+1} = \beta\left(g\left(a_y^{k+1}\right) - g\left(a_y^k\right)\right) + \beta\left(R^{2^{k+1}}D - R^{2^k}D\right) \tag{60}$$

Equation (60) represents the residual value of the dual feasibility condition. Hence, S_1^{k+1} represents the dual residual of iteration.

$$r_1^{k+1} = K_a - 2K_b a_y^{k+1} + {a_y^{k+1}}^T K a_y^{k+1} D - R^{2^{k+1}}D - u^{k+1} \tag{61}$$

$$r_2^{k+1} = 1 - {\eta^{k+1}}^T D \tag{62}$$

$$r_3^{k+1} = \left(2D{a_y^{k+1}}^T K - 2{K_b}^T\right){\eta^{k+1}}^T \tag{63}$$

Equations (61)–(63) represent the primal residuals of iteration. The KKT conditions of problem (32) consist of four components, corresponding to the primal and dual residuals. These two types of residuals gradually converge to zero with the use of the fast ADMM algorithm.

Both the primal and dual residuals must be sufficiently small, meeting the conditions presented below, for the termination of the use of the fast ADMM algorithm:

$$\max\left\{\begin{array}{l} \frac{\|r_1^k\|}{\max\left\{\|2K_b a_y + a_y^T K a_y D - R^2 D\|, \|u\|, \|K_a\|\right\}}, \frac{\|S_1^k\|}{\|{\eta^k}^T\|}, \frac{\|r_2^k\|}{\max\left\{1, \|{\eta^k}^T D\|\right\}}, \\ \frac{\|r_3^k\|}{\max\left\{\|2D{a_y^{k+1}}^T K - 2{K_b}^T\|, \|2{\eta^{k+1}}^T\|\right\}} \end{array}\right\} \leq \varepsilon^{tol} \tag{64}$$

where $\varepsilon^{tol} > 0$. Typically, ε^{tol} is selected so that $10^{-4} \leq \varepsilon^{tol} \leq 10^{-3}$; in this study, $\varepsilon^{tol} = 10^{-3}$. This ensures that the algorithm terminates when the solution's accuracy is within an acceptable range.

6. Experiment

In this section, we describe extensive experiments conducted on various datasets to validate the effectiveness and robustness of the three truncated loss function SVDD algorithms proposed in this study, which are presented below. The experimental datasets consisted of synthetic and several UCI datasets. We also compared hinge loss SVDD [8], DW-SVDD [20], R-SVDD [19], and GL-SVDD [16] algorithms to validate their performance.

1. ϕ_R-SVDD: SVDD algorithm with a truncated generalized ramp loss function;
2. ϕ_f-SVDD: SVDD algorithm with a truncated binary cross entropy loss function;
3. ϕ_L-SVDD: SVDD algorithm with a truncated linear exponential loss function.

6.1. Experimental Setup

This subsection introduces the evaluation metrics, kernels, and parameter settings used in the experiments.

6.1.1. Evaluation Metrics

We used G-mean and F_1-score as evaluation metrics to assess the performance of the different methods described in this study [30,39–41]. They are based on TP, FN, FP, and TN, where TP denotes the number of target data predicted as target data, FN denotes the number of target data predicted as outlier data, FP denotes the number of outlier data predicted as target data, and TN denotes the number of outlier data predicted as outlier data.

$$G - mean = \sqrt{\text{Recall} \times Specificity} \tag{65}$$

$$F_1 - score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{66}$$

where $Recall = TP/(TP + FN)$, $Specificity = TN/(TN + FP)$, and $Precision = TP/(TP + FP)$. It can be seen from (65) and (66) that G-mean can provide a good balance between recall and specificity, and F_1-score provides a good balance between precision and recall. Hence, both of them can measure the performance of the proposed method, comprehensively.

6.1.2. Kernels

The Gaussian kernel function has the best mapping ability and is the most practical method, commonly used to handle the classification problems of nonlinear data, and we used it when conducting our experiments. The definition of the Gaussian kernel function is as follows:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{67}$$

6.1.3. Parameter Configuration

To ensure a fair comparison, the parameters for each method were selected using the grid search method. For all methods, the regularization parameter c was selected from $\{0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1, 2, 5, 10\}$. Arin Chaudhuri [42] has demonstrated that there exists a σ in the interval $\left[\min\left\{\sqrt{\|x_i - x_j\|^2}, i \neq j\right\}/\sqrt{3}, \max\left\{\sqrt{\|x_i - x_j\|^2}, i \neq j\right\}/\sqrt{3}\right]$ that maximizes the SVDD optimization objective function. Therefore, the range for the kernel parameter σ can be set as: $\left[\min\left\{\sqrt{\|x_i - x_j\|^2}, i \neq j\right\}/\sqrt{3},\ \max\left\{\sqrt{\|x_i - x_j\|^2}, i \neq j\right\}/\sqrt{3}\right]$.

The DW-SVDD and GL-SVDD algorithms locate the k-nearest neighbors in the feature space, with the parameter k selected from $\{5, 10, 15, 20\}$. All of the truncated loss function SVDD algorithms select the parameter λ from $\{0.1, 0.2, 0.5, 1\}$, ϕ_R-SVDD selects parameter v from $\{0.1, 0.3, 0.5, 1, 5\}$, and ϕ_f-SVDD selects parameter θ from $\{0.1, 0.3, 0.5, 1, 5\}$. The parameter a for ϕ_L-SVDD is selected from $\{1, 5, 10\}$.

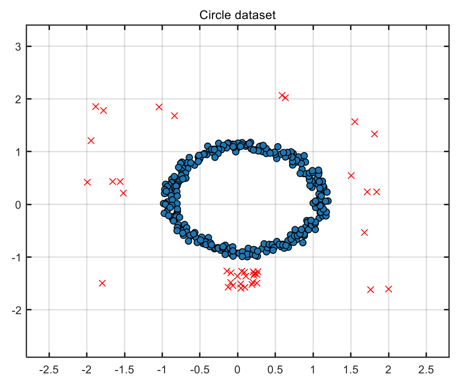
6.2. Synthetic Datasets with Noise

In this study, we constructed two synthetic datasets: circular and banana-shaped, to test the robustness of the three proposed truncated loss function support vector data description (SVDD) algorithms. To verify the robustness of the algorithms in handling different types of noise, we introduced two types of noise:

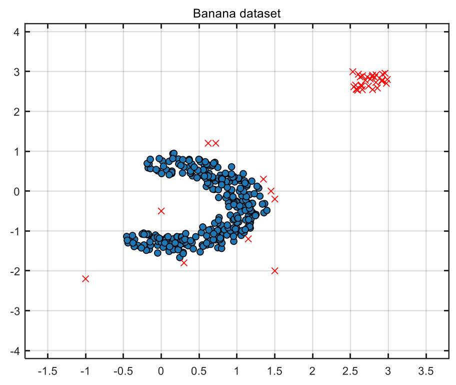
- Neighboring noise: Noise points are randomly distributed near the normal samples but do not completely overlap with the normal samples, forming a more discrete distribution characteristic. This noise simulates the common boundary ambiguity in practical applications.
- Regional noise: Noise points are randomly distributed within a specified area, forming sparse clusters. This noise simulates possible local anomalies in practical applications.

The description of the synthetic dataset is given below:

1. Circular Dataset
    - Normal samples: This consists of 300 two-dimensional sample points distributed within a concentric circle, forming a normal distribution pattern.
    - Noise samples: This consists of 40 noise points, divided into two types: 20 noise points randomly distributed near the normal sample points, showing a discrete distribution characteristic, and another 20 noise points randomly distributed within a specified area, forming sparse clusters.
    - Illustration: Figure 1a shows the circular dataset, where blue points represent normal samples, and red points represent noise.



(**a**)



(**b**)

Figure 1. Two synthetic datasets, where blue samples are normal and the samples marked with a red cross represent noise. (**a**) Circular dataset. (**b**) Banana-shaped dataset.

2. Banana-shaped Dataset
    - Normal samples: This consists of 300 two-dimensional sample points distributed along curved lines, resembling a banana shape.
    - Noise samples: This consists of 40 noise points, divided into two types: 10 noise points randomly distributed near the banana-shaped normal sample points, and another 30 noise points randomly distributed within a specified area, forming sparse clusters.
    - Illustration: Figure 1b shows the banana-shaped dataset, where blue points represent normal samples, and red points represent noise samples.

By choosing these noise distributions and quantities, we aimed to simulate different real-world noise scenarios and test the robustness of the proposed algorithms under varying noise conditions.

For circular outliers, the classification results of SVDD, DW-SVDD, DBSCAN, ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD algorithms for this dataset are presented in Figure 2. We can observe that the SVDD algorithm accurately identifies seven noise points as outliers, while misclassifying thirty-three noise points as normal values. Therefore, the performance of SVDD is severely affected by noise. Unlike SVDD, DW-SVDD accurately identified 22 noise points as outliers, while misclassifying 18 clustered noise points as normal values. DW-SVDD is based on a weighted idea where outliers are commonly sparse data; hence, a smaller weight is designated to these sparse data, which are then excluded from the sphere. The weighted method struggles to exclude clustered noise since 20 clustered noise points are present in the circular outliers. DBSCAN accurately identified the 20 sparsely distributed noise points but incorrectly classified the 20 clustered noise points as normal values. Therefore, DBSCAN cannot accurately identify clustered noise. The ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD algorithms successfully identified all of the noise points, and their classification boundaries encompassed all of the target samples.

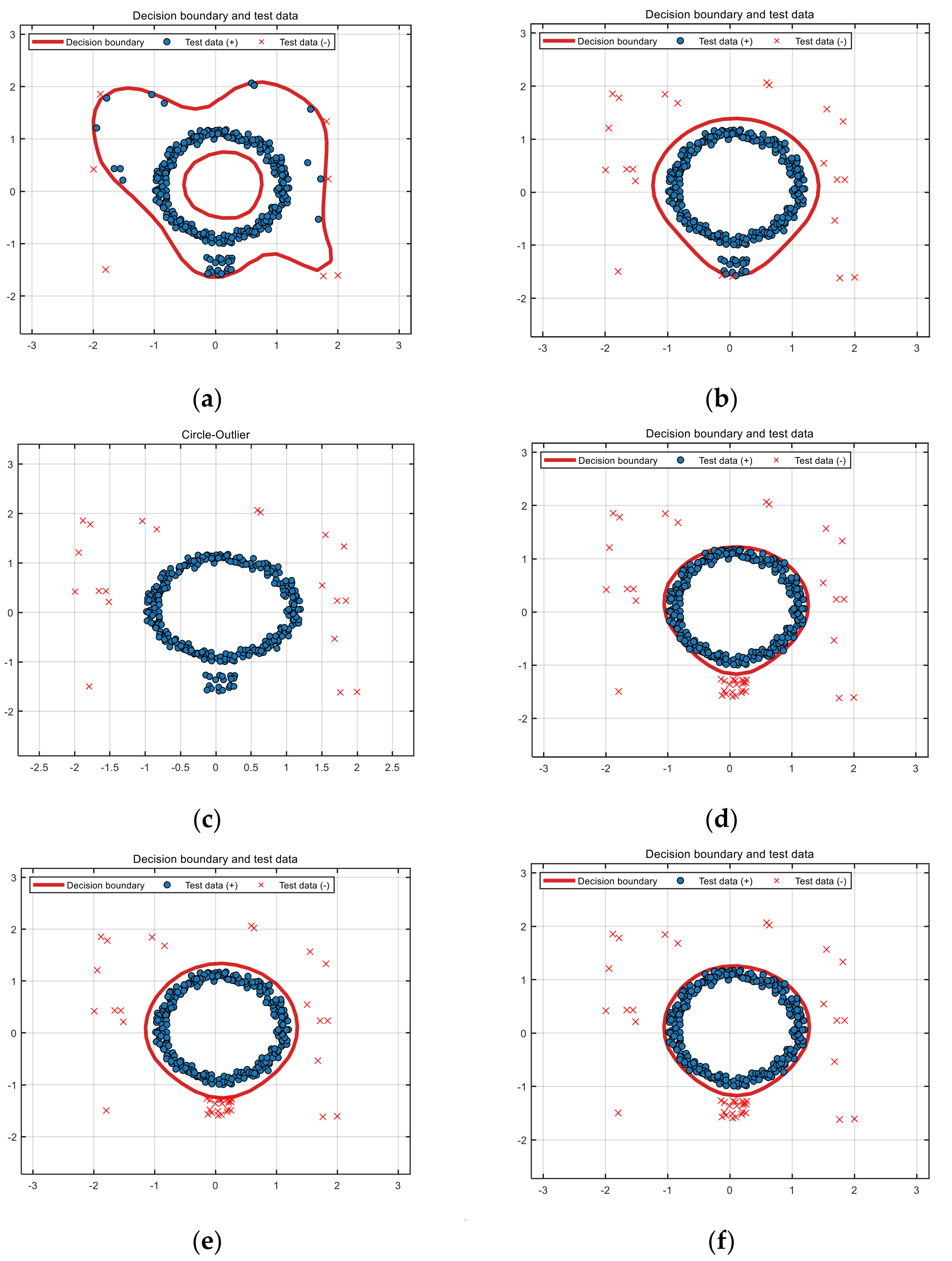


Figure 2. Classification results of six methods for circular dataset: (**a**) SVDD, (**b**) DW-SVDD, (**c**) DBSCAN, (**d**) ϕ_R-SVDD, (**e**) ϕ_f-SVDD, and (**f**) ϕ_L-SVDD.

The classification results of the SVDD, DW-SVDD, DBSCAN, ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD algorithms for the banana-shaped dataset are presented in Figure 3. It can be observed that SVDD only identifies two noise points, while misclassifying the remaining noise points as normal values. DW-SVDD accurately differentiated nine noise points distributed around the banana data, while misclassifying the remaining noise points as normal values. ϕ_R-SVDD accurately identified 35 noise points as outliers, while misclassifying the remaining noise points as normal values, and erroneously classifying 16 target values as outliers. ϕ_f-SVDD accurately identified 36 noise points as outliers, while misclassifying the remaining noise points as normal values, and erroneously classifying 14 target values as outliers. ϕ_L-SVDD accurately identified 38 noise points as outliers, while misclassifying the remaining noise points as normal values, and erroneously classifying 13 target values as outliers.

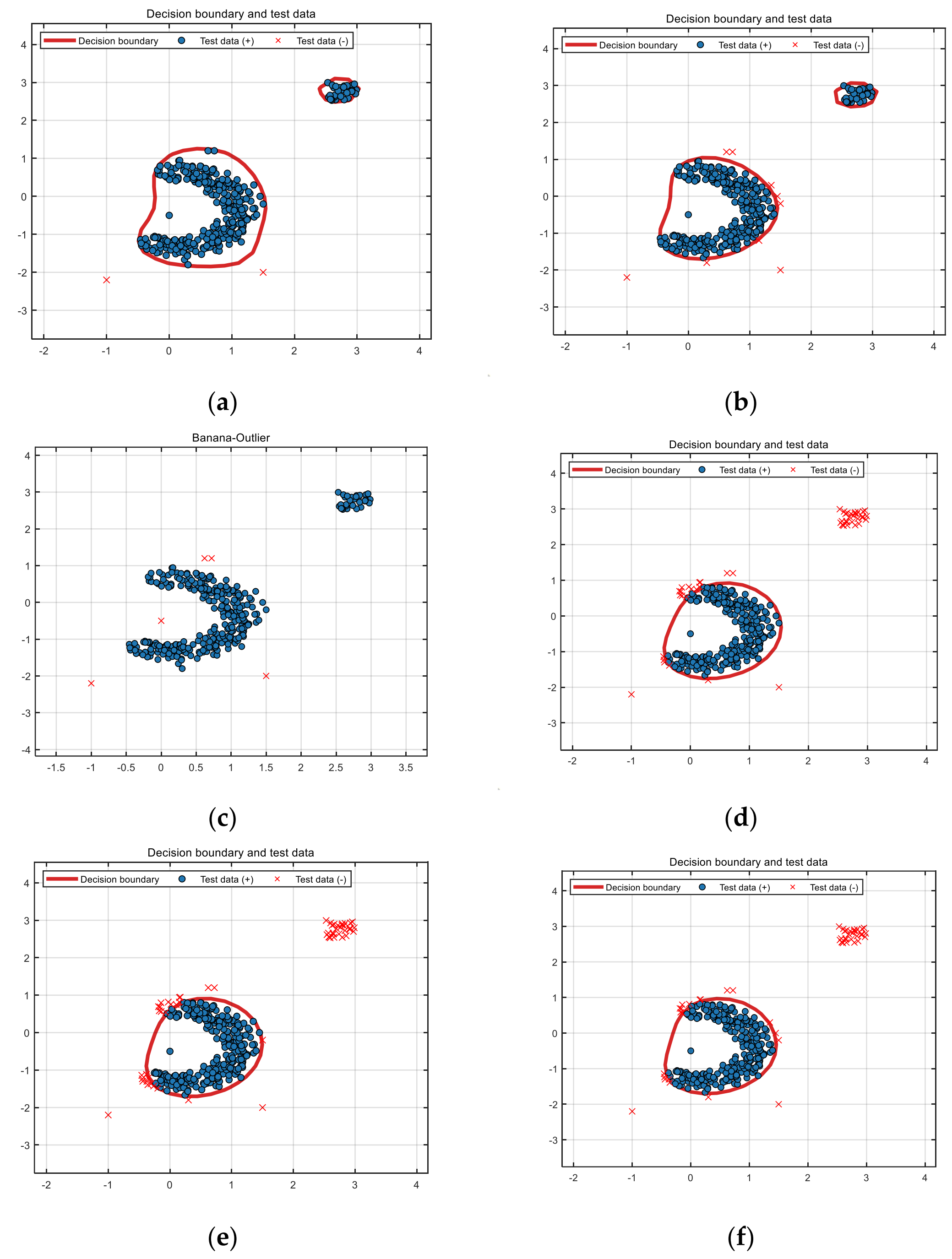


Figure 3. Classification results of six methods for banana-shaped dataset: (**a**) SVDD, (**b**) DW-SVDD, (**c**) DBSCAN, (**d**) ϕ_R-SVDD, (**e**) ϕ_f-SVDD, and (**f**) ϕ_L-SVDD.

From the experiments on the two synthetic datasets, it can be seen that the DBSCAN algorithm is very effective in handling obvious noise, but it cannot accurately identify noise when the noise is less obvious or close to normal data. DW-SVDD and DBSCAN methods cannot accurately identify sparse clustered noise, whereas ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD can effectively identify clustered noise in the dataset. Figures 1 and 2 show that the classification boundaries of ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD are tighter and smoother than SVDD and DW-SVDD methods. Therefore, it can be determined that the three truncated loss function SVDD algorithms proposed in this study are more robust to noise in both synthetic datasets.

6.3. UCI Datasets with the Presence of Noise

To further validate the effectiveness of the proposed method, we used eight standard datasets obtained from the UCI machine learning repository to conduct our experiments [43]. For each standard dataset, one class of samples was used as normal data, and the remaining classes were used as outlier data. Moreover, to eliminate the impact of data scale, each feature of every dataset was normalized to $[0, 1]$. Grid parameter optimization was used to determine the optimal parameters in all of the experiments, and the average value of ten repetitions of each experiment was the final result. For each dataset, parts of the target and non-target data were randomly selected for the training set, and the remaining target and non-target data were selected for the test set.

6.3.1. Training Dataset without Non-Target Data

For each dataset, 70% of the normal data were randomly selected for the training set, and noise-contaminated data were added to the training set. Noise data were created by changing the labels of non-target data from -1 to 1, with added noise data proportions of 10%, 20%, 30%, 40%, and 50% non-target data, and the test set consisted of the remaining target and non-target data.

Table 1 presents the G-mean averages for different methods across 10 trials, with the best results presented in bold. In 40 experimental datasets, ϕ_R-SVDD and ϕ_f-SVDD achieved higher G-means on 35 datasets compared to benchmark methods, while ϕ_L-SVDD achieved higher G-means on 34 datasets. As the proportion of noise-contaminated data increases, i.e., more data are contaminated by noise, the classification accuracy of all models generally declines. It can be observed that, as the proportion of noise-contaminated data increases, the ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD algorithms show less of a decline in accuracy compared to the other algorithms. Taking the Iris dataset as an example, when the proportion of noise-contaminated data r increases from 0.1 to 0.5, the G-means of ϕ_R-SVDD,ϕ_f-SVDD, and ϕ_L-SVDD decline by 13.02%, 13.02%, and 11.5%, while those of the DW-SVDD, R-SVDD, and GLE-SVDD decline by 19.55%, 16.6%, and 17.7%, respectively.

Table 2 shows the F_1-score averages of different methods over 10 trials. In the 40 experimental datasets, ϕ_R-SVDD and ϕ_f-SVDD achieved higher F_1-scores than the compared methods in 32 datasets, while ϕ_L-SVDD had higher F_1-scores in 31 datasets. ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD demonstrate higher classification accuracy in most test datasets compared to current noise-resistant SVDD models (i.e., DW-SVDD, R-SVDD, and GL-SVDD), indicating that the use of a truncated loss function makes SVDD less sensitive to noise.

Table 1. G-mean values for different methods for UCI datasets with noise (not containing non-target data).

Dataset	r	SVDD	DW-SVDD	R-SVDD	GL-SVDD	ϕ_R-SVDD	ϕ_f-SVDD	ϕ_L-SVDD
Blood	10%	49.95	49.28	51.25	49.19	**52.79**	**52.79**	**54.82**
	20%	46.87	46.43	48.65	46.84	**50.21**	**50.21**	**51.5**
	30%	42.34	41.87	42.34	42.34	**45.96**	**45.96**	**52.41**
	40%	41.39	41.52	41.39	41.58	**45.06**	**45.06**	**51.50**
	50%	40.53	40.56	40.69	40.9	**42.6**	**42.6**	**50.72**
Balance scale	10%	73.70	73.8	73.91	73.64	**74.38**	**74.38**	**74.32**
	20%	**68.58**	**68.67**	**68.60**	**68.60**	66.62	66.62	67.19
	30%	62.66	**64.26**	62.66	62.71	62.70	62.70	63.75
	40%	57.38	58.44	57.38	57.44	**58.60**	**58.60**	**64.32**
	50%	53.47	54.72	53.43	53.47	**56.51**	**56.51**	**61.13**
Ecoli	10%	77.5	77.48	80.63	76.28	**86.08**	**86.08**	**81.95**
	20%	70.85	67.02	70.26	70.85	**73.08**	**73.08**	**73.53**
	30%	67	62.77	67.22	67	**70.61**	**70.61**	**70.29**
	40%	64.75	60.37	64.75	64.75	**67.57**	**67.57**	**68.33**
	50%	59.94	55.18	59.87	59.94	**64.26**	**64.26**	**64.42**
Haberman	10%	**54.37**	**54.57**	53.36	**55.49**	54.06	54.06	52.09
	20%	54.85	55.01	54.87	54.93	**55.34**	**55.34**	**54.30**
	30%	53.38	53.58	54.86	53.54	**57.13**	**57.13**	**56.02**
	40%	52.54	52.31	52.73	52.45	**53.89**	**53.89**	**53.93**
	50%	50.12	49.74	49.95	50.09	**51.92**	**51.92**	**52.03**
Iris	10%	76.8	79.5	76.92	78.18	**80.74**	**80.74**	**79.52**
	20%	69.29	68.94	69.39	69.52	**71.13**	**71.13**	**70.97**
	30%	65.77	65.52	65.67	65.85	**70.46**	**70.46**	**69.46**
	40%	61.67	61.69	61.41	61.96	**65.46**	**65.46**	**65.84**
	50%	60.21	59.95	60.32	60.48	**67.72**	**67.72**	**68.02**
Wine	10%	79.76	79.76	79.02	79.76	**80.18**	**80.18**	**80.74**
	20%	71.66	73.59	73.02	72.32	**75.03**	**75.03**	**76.23**
	30%	68.73	70.11	71.49	69.04	**73.12**	**73.12**	**72.44**
	40%	64.3	64.25	64.21	64.01	**68.35**	**68.35**	**68.96**
	50%	64.03	63.93	64.06	63.09	**68.00**	**68.00**	**68.59**
Ionosphere	10%	84.22	84.43	85.08	84.22	**86.89**	**86.89**	**86.81**
	20%	82.98	83.3	82.89	83.22	**84.12**	**84.12**	**83.38**
	30%	81.94	82.2	81.35	81.89	**82.42**	**82.42**	**82.18**
	40%	**83.55**	**83.37**	80.58	**83.33**	83.14	83.14	82.86
	50%	**82.28**	**82.44**	77.87	**82.28**	81.96	81.96	82.06
Sonar	10%	56.46	56.89	56.16	57.22	**58.62**	**58.62**	**59.58**
	20%	53.46	51.48	53.46	53.46	**53.50**	**53.50**	**53.73**
	30%	49.15	47.17	49.15	49.15	**55.29**	**55.29**	**55.28**
	40%	52.48	50.04	52.48	52.48	**57.57**	**57.57**	**57.68**
	50%	45.22	43.31	45.22	45.22	**50.10**	**50.10**	**50.38**

Table 2. F_1-score values of different methods for UCI datasets with noise (not containing non-target data).

Dataset	r	SVDD	DW-SVDD	R-SVDD	GL-SVDD	ϕ_R-SVDD	ϕ_f-SVDD	ϕ_L-SVDD
Blood	10%	72.17	**73.13**	**73.18**	**73.21**	73.05	73.05	72.93
	20%	**68.98**	68.09	**69.12**	**68.85**	68.72	68.72	68.19
	30%	65.93	66.54	67.33	65.93	**68.42**	**68.42**	**68.44**
	40%	64.42	64.95	64.26	64.28	**66.25**	**66.25**	**65.95**
	50%	63.38	62.37	63.27	62.64	**65.25**	**65.25**	**64.06**
Balance scale	10%	**58.14**	**58.28**	**58.13**	**58.07**	56.67	56.48	57.01
	20%	50.57	**51.47**	50.19	**51.49**	50.74	50.74	51.14
	30%	48.61	48.92	48.63	48.65	**49.75**	**49.75**	**49.58**
	40%	48.18	48.82	48.18	48.21	**53.02**	**53.02**	**53.65**
	50%	49.07	49.21	49.16	49.21	**49.81**	**49.81**	**53.48**
Ecoli	10%	60.55	60.48	60.51	55.27	**72.52**	**63.25**	**64.12**
	20%	47.43	50.81	46.49	50.81	**54.71**	**54.60**	**56.57**
	30%	49.30	47.04	43.51	49.30	**53.46**	**53.26**	**53.27**
	40%	50.95	49.43	46.08	50.95	**53.02**	**53.02**	**53.65**
	50%	50.49	50.58	49.20	50.58	**51.93**	**51.93**	**52.11**
Haberman	10%	57.98	58.28	56.01	**62.42**	57.63	58.15	57.79
	20%	**60.97**	60.17	59.67	60.29	60.65	60.27	59.76
	30%	62.18	61.95	62.36	62.23	**64.10**	**63.72**	**64.22**
	40%	63.87	64.41	62.04	64.02	**64.62**	**64.45**	**63.87**
	50%	56.66	58.02	57.51	58.25	**60.52**	**60.61**	**59.11**
Iris	10%	**66.07**	57.62	**68.75**	56.52	60.02	60.41	62.75
	20%	51.17	51.71	53.99	51.79	**55.66**	**55.66**	**55.79**
	30%	41.10	41.02	42.15	41.17	**49.74**	**49.74**	**51.22**
	40%	40.04	40.13	41.97	40.41	**44.94**	**44.94**	**45.79**
	50%	43.39	43.70	44.28	43.99	**49.97**	**49.97**	**50.34**
Wine	10%	60.87	63.87	62.77	53.66	**66.26**	**66.76**	**67.28**
	20%	44.97	43.48	41.97	43.94	**55.32**	**50.34**	**51.50**
	30%	42.46	43.91	43.33	42.81	**48.67**	**47.52**	**48.27**
	40%	42.06	42.45	42.56	41.94	**45.60**	**45.29**	**49.26**
	50%	46.64	46.61	46.24	45.98	**48.96**	**48.94**	**49.59**
Ionosphere	10%	80.87	81.09	81.29	80.87	**83.08**	**82.89**	**83.08**
	20%	80.40	80.03	80.74	80.31	**81.00**	**80.36**	**80.28**
	30%	79.46	79.76	79.98	79.40	**81.21**	**80.60**	**80.52**
	40%	81.52	81.93	81.26	81.88	**82.00**	**82.12**	**82.02**
	50%	81.85	81.67	81.48	81.67	**82.08**	**81.99**	**81.97**
Sonar	10%	**55.23**	54.3	**55.2**	54.6	54.17	54.17	55.14
	20%	51.43	52.25	52.25	52.35	**53.03**	**52.73**	52.13
	30%	48.73	47.78	48.73	48.73	**51.52**	**51.52**	**51.47**
	40%	51.16	50.34	51.16	51.16	**52.29**	**52.29**	**52.30**
	50%	48.26	49.08	49.08	49.08	**51.78**	**51.78**	**51.67**

6.3.2. Training Datasets with Non-Target Data

For each dataset, the training set was randomly selected to consist of 70% target and 20% non-target data, with added proportions of 10%, 20%, 30%, 40%, and 50% non-target data as noise data, and the test set included the remaining target and non-target data.

Table 3 presents the G-means from 10 trials for different methods, with the best results presented in bold. In 40 experimental datasets, ϕ_R-SVDD and ϕ_f-SVDD achieved higher G-means on 35 datasets compared to the benchmark methods, while ϕ_L-SVDD achieved a higher G-mean on 33 datasets. As the data contaminated by noise increase, the ϕ_R-SVDD, ϕ_f-SVDD, and ϕ_L-SVDD algorithms exhibit less of a decline in accuracy than the other algorithms. ϕ_R-SVDD,ϕ_f-SVDD, and ϕ_L-SVDD present almost no decline in G-means for datasets such as Blood, Iris, and Haberman.

Table 3. G-mean values of different methods on UCI datasets with noise (containing non-target data).

Dataset	*r*	SVDD	DW-SVDD	R-SVDD	GL-SVDD	ϕ_R-SVDD	ϕ_f-SVDD	ϕ_L-SVDD
Blood	10%	53.18	51.66	52.05	53.66	52.29	52.29	**54.45**
	20%	50.82	50.91	51.65	51.57	**52.02**	**52.02**	**53.39**
	30%	49.72	49.19	51.13	48.78	**53.43**	**53.43**	**52.13**
	40%	51.00	49.23	50.16	48.43	**51.21**	**51.21**	**53.87**
	50%	48.04	48.72	47.89	46.88	**52.87**	**52.87**	**52.42**
Balance scale	10%	78.86	80.91	79.7	80.24	**81.25**	**81.25**	77.64
	20%	73.53	75.25	74.18	73.41	**76.07**	**76.07**	73.65
	30%	65.78	67.38	66.2	65.51	**68.09**	**68.09**	66.48
	40%	60.74	62.30	62.10	60.56	**67.11**	**67.11**	**65.78**
	50%	56.79	61.48	58.46	60.08	**68.24**	**68.24**	**64.86**
Ecoli	10%	82.1	82.36	83.95	83.08	**85.34**	**85.34**	**82.1**
	20%	77.46	77.48	80.02	77.08	**82.53**	**82.53**	**77.82**
	30%	69.55	67.74	68.52	70.88	**71.96**	**71.96**	**71.32**
	40%	74.09	70.07	71.19	72.08	**74.45**	**74.45**	**74.35**
	50%	66.93	63.38	64.87	63.58	**68.18**	**68.18**	**67.36**
Haberman	10%	55.59	55.41	54.81	55.51	**56.72**	**56.72**	**57.09**
	20%	58.66	58.88	54.12	58.80	**61.99**	**61.99**	**59.75**
	30%	55.13	55.22	54.2	55.47	**56.14**	**56.14**	**54.65**
	40%	51.83	52.00	57.63	50.38	**56.01**	**56.01**	**54.81**
	50%	55.64	55.70	55.26	55.52	**58.27**	**58.27**	**57.60**
Iris	10%	**80.7**	**81.45**	**81.22**	**80.38**	79.13	79.13	80.18
	20%	**79.01**	**78.31**	**78.75**	**77.94**	77.37	77.37	78.63
	30%	66.77	64.48	63.58	66.63	**71.12**	**71.12**	**70.78**
	40%	67.08	60.61	63.16	64.29	**70.91**	**70.91**	**70.47**
	50%	62.19	53.74	61.09	59.39	**67.59**	**67.59**	**67.63**
Wine	10%	81.54	84.11	83.77	81.54	**84.35**	**84.35**	**82.45**
	20%	79.12	81.44	80.24	79.12	**82.87**	**82.87**	**80.80**
	30%	71.03	69.15	70.02	71.03	**75.48**	**75.48**	**71.95**
	40%	69.28	67.97	67.75	69.28	**72.29**	**72.29**	**70.47**
	50%	64.77	66.80	66.46	64.77	**69.04**	**69.04**	**66.50**
Ionosphere	10%	86.45	86.79	88.66	87.26	**89.28**	**89.28**	**89.02**
	20%	84.84	86.01	86.38	86.38	**88.49**	**88.49**	**88.28**
	30%	84.25	85.0	85.09	85.18	**87.93**	**87.93**	**87.87**
	40%	82.58	83.13	83.37	83.37	**86.60**	**86.60**	**86.20**
	50%	77.26	78.56	78.54	78.62	**84.29**	**84.29**	**83.95**
Sonar	10%	57.24	58.34	**58.97**	**58.97**	58.66	58.66	58.70
	20%	59.23	59.25	**60.35**	**60.35**	60.21	60.21	59.79
	30%	54.51	54.76	56.03	56.03	**56.50**	**56.50**	**56.33**
	40%	55.86	55.86	58.93	58.93	**58.56**	**58.56**	**60.09**
	50%	52.21	53.14	53.92	53.92	**55.45**	**55.45**	**54.71**

Table 4 shows the F_1-score averages of different methods over 10 trials. In the 40 experimental datasets, ϕ_R-SVDD achieved higher F_1-scores than the compared methods in 35 datasets, ϕ_f-SVDD performed better in 32 datasets, and ϕ_L-SVDD in 31 datasets. From Tables 1–4, it is evident that -SVDD, -SVDD, and -SVDD achieve better experimental performance than the other four methods, demonstrating superior noise resistance due to the use of the truncated loss function.

Table 4. F_1-score values of different methods for UCI datasets with noise (containing non-target data).

Dataset	r	SVDD	DW-SVDD	R-SVDD	GL-SVDD	ϕ_R-SVDD	ϕ_f-SVDD	ϕ_L-SVDD
Blood	10%	76.30	76.05	76.38	**77.20**	77.14	75.81	69.71
	20%	65.90	65.46	66.12	66.22	**73.71**	**70.79**	65.62
	30%	65.56	67.43	65.40	69.15	**69.32**	66.81	**68.44**
	40%	63.94	65.87	60.48	65.82	**70.09**	**66.07**	**66.46**
	50%	65.98	65.10	63.16	68.44	**80.57**	**68.88**	**69.03**
Balance scale	10%	65.77	68.66	66.89	67.52	**69.18**	66.75	66.74
	20%	63.48	62.23	62.34	61.64	**64.09**	61.90	**63.58**
	30%	58.62	58.54	56.99	61.23	**61.62**	**61.62**	59.06
	40%	64.74	63.67	63.67	63.16	**66.57**	**66.14**	**66.05**
	50%	68.25	68.78	66.92	68.02	**71.23**	**71.23**	**69.84**
Ecoli	10%	67.58	68.46	68.29	68.51	**76.37**	**68.84**	**68.58**
	20%	62.76	63.36	63.03	62.17	**70.01**	**69.01**	**68.16**
	30%	58.62	58.54	56.99	60.25	**61.39**	**61.75**	**61.11**
	40%	65.01	65.49	63.81	66.17	**67.89**	**67.86**	**67.67**
	50%	65.11	65.83	65.92	66.02	**66.86**	**66.17**	**67.92**
Haberman	10%	**61.22**	**61.07**	59.24	**61.12**	58.04	60.43	58.08
	20%	**66.36**	**66.28**	58.27	**66.19**	63.82	63.82	59.85
	30%	61.84	59.14	59.53	61.53	**64.97**	**65.48**	**64.48**
	40%	57.60	58.29	57.46	57.45	**63.08**	**64.17**	**62.35**
	50%	62.90	62.45	55.28	63.08	**66.34**	**70.63**	**63.96**
Iris	10%	61.66	62.84	63.26	61.13	61.91	**63.78**	60.98
	20%	55.14	55.39	56.23	53.85	**58.24**	**58.24**	**59.73**
	30%	52.68	50.74	53.21	52.68	**58.91**	**53.91**	**53.75**
	40%	54.74	51.56	53.88	55.98	**57.64**	**57.64**	**57.42**
	50%	54.19	57.43	58.42	58.47	**60.52**	**58.79**	**58.64**
Wine	10%	65.03	70.62	70.33	65.03	**70.90**	**68.82**	67.48
	20%	64.02	59.12	64.69	59.12	**62.37**	**59.29**	**59.35**
	30%	52.68	50.74	52.21	52.68	**58.91**	**53.91**	**53.75**
	40%	55.56	53.86	53.82	55.56	**58.78**	**56.90**	**56.17**
	50%	58.67	57.47	57.24	57.47	**60.52**	**59.79**	**58.94**
Ionosphere	10%	85.21	85.55	87.22	86.02	**87.95**	**87.95**	**87.64**
	20%	85.36	84.20	85.40	85.58	**87.48**	**87.48**	**87.22**
	30%	86.43	86.80	86.79	86.91	**88.45**	**88.45**	**88.38**
	40%	87.39	87.46	87.19	87.58	**88.95**	**88.8**	**88.67**
	50%	87.94	87.63	87.90	87.93	**89.24**	**89.07**	**88.92**
Sonar	10%	58.54	58.75	58.71	58.71	**59.05**	**58.88**	**59.12**
	20%	57.46	58.21	58.05	58.05	**58.74**	**58.74**	**58.18**
	30%	55.12	52.25	52.66	52.66	**60.26**	**59.52**	**56.25**
	40%	62.15	58.30	60.23	60.23	**63.52**	**63.52**	**62.38**
	50%	61.35	**64.91**	60.10	60.10	62.80	62.80	61.94

7. Conclusions

This study aimed to enhance the robustness and effectiveness of the SVDD algorithm. We propose a general framework for the truncated loss function of this algorithm, which uses bounded loss functions to mitigate the impact of outliers. Due to the non-differentiability of the truncated loss function, we employed the fast ADMM algorithm to solve the SVDD model with the truncated loss function, which handles truncated loss functions within a unified framework. In this context, the truncated generalized ramp, truncated binary cross entropy, and truncated linear exponential loss functions for the SVDD algorithm were constructed, and extensive experiments show that these three SVDD models exhibit more robustness than other SVDD models in most cases. However, this method still has the following shortcomings. Firstly, introducing the truncated loss function increases the complexity of model training, as some truncated loss functions cannot directly provide explicit expressions for neighboring point operators, requiring additional computational overhead. These factors may limit the application of the method to large-scale datasets. To overcome these limitations, future work can consider using a distributed computing framework to accelerate the training process of the ADMM algorithm. Secondly, the truncated loss function SVDD introduces new free parameters, which increases the time required for grid search parameter selection. When the data scale is large, the computation time for the grid search method may become unacceptable. To address this drawback, algorithms such as Bayesian optimization can be considered in the future to find the optimal parameters, further improving model performance and optimization efficiency. Finally, for extremely noisy data, the truncated loss function may not completely eliminate its impact, and the effect is limited. In this case, methods combining clustering algorithms such as DBSCAN can be adopted. First, clustering algorithms like DBSCAN can be used to preprocess the data and remove noise, and then the proposed method can be used to detect anomalies.

Author Contributions: H.C.: conceptualization, methodology, software, and writing—original draft preparation. Y.L.: conceptualization, resources, writing—review and editing, supervision, and funding acquisition. J.S.: conceptualization, resources, writing—review and editing, and funding acquisition. W.Z.: conceptualization, methodology, formal analysis, and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 62373301 and 62173277; the Natural Science Foundation of Shaanxi Province, grant number 2023-JC-YB-526; the Aeronautical Science Foundation of China, grant number 20220058053002; and the Shaanxi Province Key Laboratory of Flight Control and Simulation Technology.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: This work is supported by the Shaanxi Province Key Laboratory of Flight Control and Simulation Technology.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 15. [CrossRef]
2. Pimentel, M.A.; Clifton, D.A.; Clifton, L.; Tarassenko, L. A review of novelty detection. *Signal Process.* **2014**, *99*, 215–249. [CrossRef]
3. Lei, Y.; Yang, B.; Jiang, X.; Jia, F.; Li, N.; Nandi, A.K. Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mech. Syst. Signal Process.* **2020**, *138*, 106587. [CrossRef]
4. Hasani, R.; Wang, G.; Grosu, R. A machine learning suite for machine components' health-monitoring. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 9472–9477.
5. Khan, S.S.; Madden, M.G. One-class classification: Taxonomy of study and review of techniques. *Knowl. Eng. Rev.* **2014**, *29*, 345–374. [CrossRef]
6. Khan, S.S.; Madden, M.G. A survey of recent trends in one class classification. In Proceedings of the 20th Annual Irish Conference on Artificial Intelligence and Cognitive Science, Dublin, Ireland, 19–21 August 2009; pp. 188–197. [CrossRef]
7. Alam, S.; Sonbhadra, S.K.; Agarwal, S.; Nagabhushan, P. One-class support vector classifiers: A survey. *Knowl.-Based Syst.* **2020**, *196*, 105754. [CrossRef]
8. Tax, D.M.J.; Duin, R.P.W. Support vector data description. *Mach. Learn.* **2004**, *54*, 45–66. [CrossRef]
9. Zheng, S. A fast iterative algorithm for support vector data description. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 1173–1187. [CrossRef]
10. Turkoz, M.; Kim, S.; Son, Y.; Jeong, M.K.; Elsayed, E.A. Generalized support vector data description for anomaly detection. *Pattern Recognit.* **2020**, *100*, 107119. [CrossRef]
11. Fong, S.; Narasimhan, S. An Unsupervised Bayesian OC-SVM Approach for Early Degradation Detection, Thresholding, and Fault Prediction in Machinery Monitoring. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 3500811. [CrossRef]
12. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. Lof: Identifying density-based local outliers. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 15–18 May 2000; pp. 93–104.
13. Zheng, L.; Hu, W.; Min, Y. Raw Wind Data Preprocessing: A Data-Mining Approach. *IEEE Trans. Sustain. Energy* **2015**, *6*, 11–19. [CrossRef]

14. Khan, S.S.; Karg, M.E.; Kulic, D.; Hoey, J. X-factor HMMs for detecting falls in the absence of fall-specific training data. In Proceedings of the Ambient Assisted Living and Daily Activities: 6th International Work-Conference, IWAAL 2014, Belfast, UK, 2–5 December 2014; pp. 1–9.
15. Andreou, C.; Karathanassi, V. Estimation of the Number of Endmembers Using Robust Outlier Detection Method. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 247–256. [CrossRef]
16. Lu, J.; Gao, Y.; Zhang, L. A novel dynamic radius support vector data description based fault diagnosis method for proton exchange membrane fuel cell systems. *Int. J. Hydrogen Energy* **2022**, *47*, 35825–35837. [CrossRef]
17. Zhao, Y.-P.; Xie, Y.-L.; Ye, Z.-F. A new dynamic radius SVDD for fault detection of aircraft engine. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104177. [CrossRef]
18. Zhu, F.; Yang, J.; Gao, C.; Xu, S.; Ye, N.; Yin, T. A weighted one-class support vector machine. *Neurocomputing* **2016**, *189*, 1–10. [CrossRef]
19. Chen, G.; Zhang, X.; Wang, Z.J. Robust support vector data description for outlier detection with noise or uncertain data. *Knowl.-Based Syst.* **2015**, *90*, 129–137. [CrossRef]
20. Cha, M.; Kim, J.S.; Baek, J.G. Density weighted support vector data description. *Expert Syst. Appl.* **2014**, *41*, 3343–3350. [CrossRef]
21. Sadeghi, R.; Hamidzadeh, J. Automatic support vector data description. *Soft Comput.* **2018**, *22*, 147–158. [CrossRef]
22. Hu, W.; Hu, T.; Wei, Y.; Lou, J.; Wang, S. Global Plus Local Jointly Regularized Support Vector Data Description for Novelty Detection. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 6602–6614. [CrossRef] [PubMed]
23. Zhao, Y.-P.; Huang, G.; Hu, Q.-K.; Li, B. An improved weighted one class support vector machine for turboshaft engine fault detection. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103796. [CrossRef]
24. Wang, K.; Lan, H. Robust support vector data description for novelty detection with contaminated data. *Eng. Appl. Artif. Intell.* **2020**, *91*, 103554. [CrossRef]
25. Xing, H.-J.; Li, L.-F. Robust least squares one-class support vector machine. *Pattern Recognit. Lett.* **2020**, *138*, 571–578. [CrossRef]
26. Xiao, Y.; Wang, H.; Xu, W. Ramp Loss based robust one-class SVM. *Pattern Recognit. Lett.* **2017**, *85*, 15–20. [CrossRef]
27. Tian, Y.; Mirzabagheri, M.; Bamakan, S.M.H. Ramp loss one-class support vector machine; A robust and effective approach to anomaly detection problems. *Neurocomputing* **2018**, *310*, 223–235. [CrossRef]
28. Xing, H.; Ji, M. Robust one-class support vector machine with rescaled hinge loss function. *Pattern Recognit.* **2018**, *84*, 152–164. [CrossRef]
29. Zhong, G.; Xiao, Y.; Liu, B.; Zhao, L.; Kong, X. Pinball loss support vector data description for outlier detection. *Appl. Intell.* **2022**, *52*, 16940–16961. [CrossRef]
30. Zheng, Y.; Wang, S.; Chen, B. Robust one-class classification with support vector data description and mixed exponential loss function. *Eng. Appl. Artif. Intell.* **2023**, *122*, 106153. [CrossRef]
31. Le Thi, H.A.; Pham Dinh, T. DC programming and DCA: Thirty years of developments. *Math. Program.* **2018**, *169*, 5–68. [CrossRef]
32. Liu, J.; Pang, J.S. Risk-based robust statistical learning by stochastic difference-of convex value-function optimization. *Oper. Res.* **2023**, *71*, 397–414. [CrossRef]
33. Yuille, A.L.; Rangarajan, A. The concave-convex procedure. *Neural Comput.* **2003**, *15*, 915–936. [CrossRef]
34. Tao, Q.; Wu, G.; Chu, D. Improving sparsity and scalability in regularized nonconvex truncated-loss learning problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 2782–2793. [CrossRef]
35. Wang, H.J.; Shao, Y.H.; Xiu, N.H. Proximal operator and optimality conditions for ramp loss SVM. *Optim. Lett.* **2022**, *16*, 999–1014. [CrossRef]
36. Gong, P.; Zhang, C.; Lu, Z. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16–21 June 2013; pp. 696–704.
37. Scholkopf, B.; Herbrich, R.; Smola, A.J. A generalized representer theorem. In Proceedings of the 14th Annual Conference on Computational Learning Theory, Amsterdam, The Netherlands, 16–19 July 2001; pp. 416–426. [CrossRef]
38. Guan, L.; Qiao, L.; Li, D.; Sun, T.; Ge, K.; Lu, X. An efficient ADMM-based algorithm to nonconvex penalized support vector machines. In Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, 17–20 November 2018; pp. 1209–1216.
39. Wu, M.; Ye, J. A small sphere and large margin approach for novelty detection using training data with outliers. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 2088–2092. [CrossRef] [PubMed]
40. Xing, H.; Liu, Y.; He, Z. Robust sparse coding for one-class classification based on correntropy and logarithmic penalty function. *Pattern Recognit.* **2021**, *111*, 107685. [CrossRef]
41. Zheng, Y.; Wang, S.; Chen, B. Multikernel correntropy based robust least squares one-class support vector machine. *Neurocomputing* **2023**, *545*, 126324. [CrossRef]

42. Chaudhuri, A.; Sadek, C.; Kakde, D.; Wang, H.; Hu, W.; Jiang, H.; Kong, S.; Liao, Y.; Peredriy, S. The trace kernel bandwidth criterion for support vector data description. *Pattern Recognit.* **2021**, *111*, 107662. [CrossRef]
43. Dua, D.; Graff, C. UCI Machine Learning Repository. 2008. University of California, Irvine, School of Information and Computer Sciences. Available online: https://archive.ics.uci.edu/ml (accessed on 20 July 2024).