AIMS Mathematics, 10(4): 8036–8065.
DOI: 10.3934/math.2025369
Received: 22 December 2024
Revised: 23 March 2025
Accepted: 27 March 2025
Published: 07 April 2025

https://www.aimspress.com/journal/Math

Research article

Adaptive robust AdaBoost-based kernel-free quadratic surface support vector machine with Universum data

Bao Ma[1], **Yanrong Ma**[2,*] **and Jun Ma**[1]

[1] School of Mathematics and Information Sciences, North Minzu University, Yinchuan Ningxia 750021, China

[2] School of Preparatory Education, North Minzu University, Yinchuan 750021, China

* **Correspondence:** Email: myr01@163.com.

Abstract: In this paper, we proposed a novel binary classification framework named adaptive robust AdaBoost-based kernel-free quadratic surface support vector machine with Universum data (A-R-U-SQSSVM). First, we developed R-U-SQSSVM by integrating the capped $L_{2,p}$-norm distance metric and the generalized Welsch adaptive loss function to improve the model's robustness and adaptability. Furthermore, we introduced Universum data points into R-U-SQSSVM to enhance the model's generalization performance by incorporating valuable prior knowledge for the classifier. Additionally, we utilized R-U-SQSSVM as a weak classifier and embedded the AdaBoost algorithm within it to obtain a strong classifier, A-R-U-SQSSVM. To effectively solve our model, we transformed it into a quadratic programming problem using the half-quadratic (HQ) optimization algorithm and concave duality. This transformed problem can be solved using convex optimization methods, such as the sequential minimal optimization (SMO) algorithm. Experimental results on University of California, Irvine (UCI) datasets demonstrated the superior classification performance of our method. In large datasets, A-R-U-SQSSVM was hundreds or even a thousand times faster than traditional capped twin support vector machine (CTSVM), SQSSVM.

Keywords: Universum; quadratic surface support vector machine; robust distance metric; AdaBoost
Mathematics Subject Classification: 68T10, 91C20

1. Introduction

Support vector machines (SVM) are a popular machine learning algorithm widely used in real-life applications such as flood prediction [1], image annotation [2], and environmental remote sensing [3]. The classical linear SVM was first proposed by Vapnik et al. [4] based on the principle of structural risk minimization for solving classification problems. Its learning strategy aims to identify the appropriate

hyperplane that maximizes the separation margin. The mathematical problem corresponding to SVM is a convex optimization problem, which can be efficiently solved by the sequential minimal optimization (SMO) algorithm [5,6]. However, traditional SVM often suffers from serious impacts on its robustness and classification performance when facing complex data distributions and datasets with noise and outliers.

In order to enhance and build robust SVM, there are currently three main approaches. The first is to replace the convex and unbounded hinge loss in SVM, the second is to construct a robust distance metric, and the third is to develop a robust estimator. In the aspect of build robust loss function: Reference [7] proposed a non-convex truncated hinge loss and solved the non-convex problem by decomposing it into several convex subproblems using differential convex optimization methods, leading to enhanced robustness. Similarly, reference [8] utilized the same technique and introduced a non-convex ramp loss function. However, they transformed the non-convex problem into a convex one using the concave-convex procedure method and then solved it using a Newton-type algorithm. Due to the nonc-onvex smoothness of the ramp loss function, it is insensitive to outliers. Reference [9] proposed a novel pinball loss function that penalizes correctly classified points. Then, to maintain sparsity in SVM, reference [10] proposed a truncated pinball loss function. However, all of the aforementioned loss functions use the L_2 norm distance metric, which may magnify the impact of outliers. Based on this, reference [11] proposed a loss function with a capped $L_{2,p}$-norm as the distance metric and achieved better robustness by adjusting the capped parameter p. Reference [12] proposed a twin SVM with a capped $L_{2,1}$-norm as the distance metric for semi-supervised classification. This model considers both robustness and discriminability. In addition, to enhance the robustness of SVM, researchers have introduced various regularization terms or utilized more robust loss functions [13,14]. In reference [13], a bounded, smooth, and non-convex Welsch loss function is proposed to enhance the robustness of semi-supervised learning (SSL) methods to outliers.

In the aspect of build robust loss estimators: Recent research findings indicate that robust loss functions based on mixed paradigms (such as hybrid ordinary-Welsch function (HOW), hybrid ordinary-L_p function (HOP) [15], and hybrid ordinary-Cauchy function (HOC) [16]) can effectively enhance the model's noise resistance. For example, the HOW proposed by Wang et al. [17] demonstrates strong robustness to outliers in low-rank matrix recovery tasks by combining the advantages of ordinary least squares and the Welsch function. Experiments have shown that HOW reduces reconstruction error by approximately 12.7% compared to the traditional Welsch function at the same noise level. Additionally, Wang et al. [18] designed a robust matrix completion framework based on truncated quadratic loss, further validating the effectiveness of the truncation mechanism in mitigating the impact of outliers. In addition, certain achievements have been made in improving SVM for robust classification through optimization techniques. For example, reference [19] explores optimization techniques suitable for robust classification, which enhance the SVM's resistance to noise and outliers to some extent, allowing the model to maintain good classification performance on some complex datasets. At the same time, there has been in-depth research on feature selection and building more robust machine learning models. Reference [20] provides profound insights in this area, indicating that reasonable feature selection can reduce the interference of noise and redundant information on the model, further enhancing the model's robustness. The relatively robust SVM and its variants have also been applied in real life, such as the medical field [21] and face recognition [22].

However, existing methods still have many limitations. On one hand, when dealing with scenarios

that include Universum data, existing models are unable to fully leverage the additional information carried by the Universum data. This makes it difficult to achieve a more comprehensive understanding and modeling of the data distribution, resulting in challenges in simultaneously achieving ideal classification accuracy and robustness in complex data environments. The concept of Universum learning was initially introduced by Vapnik [23] to acquire additional prior knowledge about data samples. For the classification problem, the Universum data does not belong to the positive class or the negative class; instead, it must fall between the hyperplanes of the two classes (as shown in Figure 1(a)). The Universum dataset contains valuable prior knowledge that is beneficial for classification. By utilizing Universum data with SVM, reference [24] has proposed Universum SVM (U-SVM). Their experiments showed that the classification performance of U-SVM, which incorporates more prior knowledge, outperforms that of SVM. Besides, the reference [25] introduced a novel boosting algorithm called U-Boost, which is based on the concept of large-margin learning and utilizes Universum data. The experimental results also demonstrate that U-Boost can provide higher classification accuracy. In recent years, the reference [26] proposes a robust kernel-free SVM based on the fuzzy membership function of the Universum. Based on Universum data and the L_1-norm, [27] proposes a model that is conducive to detecting the sparsity of the Hessian matrix of quadric surfaces. In addition, classification models constructed by utilizing Universum data with SVM and other classifiers have also been applied in real-life scenarios, such as gender classification [28], text classification [29], and the diagnosis of Alzheimer's disease [30]. On the other hand, many robust SVM methods rely on kernel functions. Although kernel functions can map data to high-dimensional spaces to solve nonlinear classification problems, the selection of kernel functions and parameter tuning is quite complex. Different kernel functions and their parameter settings can have a significant impact on model performance, and the computational complexity is relatively high. Based on this, Dagher [31] proposed a kernel-free QSSVM inspired by the concept of SVM. The main objective is to identify a quadratic separation surface that maximizes the sum of relative geometric margins. Luo et al. [32] proposed the soft-margin quadratic surface support vector machine (SQSSVM). This model penalizes all misclassified points to handle nonlinearly inseparable datasets. Later, Bai et al. [33] proposed a least-squares version of the method and applied it to classify target diseases. In recent years, Xin et al. [34] proposed an SVM classifier for handling nonlinearly separable datasets based on QSSVM by utilizing preselected techniques and feature mapping. Reference [35] proposed a kernel-free soft quadratic surface SVM model by utilizing the double-well potential function for highly nonlinear binary classification. The experiments also demonstrate the advantages of kernel-free SVM in handling nonlinear separable datasets. Nowadays, kernel-free SVM has been applied in real-life applications such as cross-selling recommendations [36] and credit risk assessment [37].

In this paper, we propose a novel framework called the adaptive robust SQSSVM with Universum data based on the AdaBoost algorithm (A-R-U-SQSSVM). This framework utilizes robust distance metric learning and Universum data learning to achieve significant robustness and classification performance. Specifically, it is designed to handle nonlinear, separable datasets. First, we are improving SQSSVM. Specifically, we replace the L_2-norm in QSSVM with a capped $L_{2,p}$-norm distance metric for improved robustness. Additionally, we replace the misclassification term in QSSVM with a generalized Welsch loss function for improved adaptability. In addition, based on SQSSVM's superior robustness and adaptability, Universum data learning is incorporated into SQSSVM to enhance its classification performance. Besides, we integrate the proposed R-U-SQSSVM

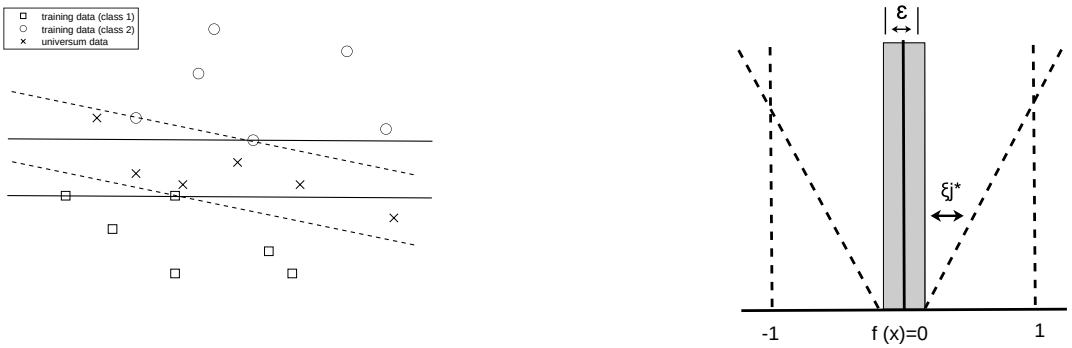
into the AdaBoost algorithm to ensemble the weak classifier with the strong SQSSVM classifiers to enhance the classification performance of A-R-U-SQSSVM (as shown in Figure 1). Table 1 shows the relationship between our model and other related works. The main contributions of this paper can be summarized as follows:

• First, we improve the loss function term and the misclassification term in SQSSVM by using a capped $L_{2,p}$-norm distance metric and a generalized Welsch loss function. Second, we introduce Universum data into R-SQSSVM. Finally, we integrate AdaBoost into the R-U-SQSSVM framework to create a robust classifier, A-R-U-SQSSVM, capable of handling nonlinear separable datasets.

• We utilize the half-quadratic (HQ) optimization algorithm and the concave duality theorem to design an efficient iterative optimization algorithm for solving our model.

• In order to test the performance of our proposed model, we conducted experiments on UCI benchmark datasets. Additionally, we conducted parameter analysis and statistical testing. The experimental results show that the proposed model demonstrates good robustness and classification performance.

The rest of the paper is organized as follows: Section 2 provides a brief review of the U-SVM and QSSVM models. Section 3 introduces our proposed model, A-R-U-SQSSVM. Section 4 is the experimental part. Section 5 concludes the paper and offers insights into future research directions.



(a) Universum data and two classes of training data. (b) The ε-insensitive loss for Universum samples.

Figure 1. Description of Universum data.

Table 1. Summary of relevant important models.

Reference	Model	Kernel
Vapnik et al. (1995) [4]	Linear SVM	Kernel-based
Weston et al. (2006) [24]	Universum SVM	Kernel-based
Dagher (2008) [31]	QSSVM	Kernel-free
Luo et al. (2016) [32]	SQSSVM	Kernel-free
Yuan et al. (2021) [11]	Capped $l_{2,p}$-norm least square twin SVM	Kernel-based
Zhang et al. (2022) [12]	Capped l_1-norm twin bounded SVM	Kernel-based
Yan et al. (2023) [26]	Fuzzy Universum SQSSVM	Kernel-free
Hossein et al. (2023) [27]	L_1-norm Universum SQSSVM	Kernel-free

2. Related works

In this sectionwe will first describe the notations used throughout the paper. Then, we will provide a brief review of the U-SVM [24], the QSSVM [31], and its soft-margin version.

2.1. Notations

We consider a binary classification task in n-dimensional Euclidean space $\mathbb{R}$, where the training dataset is represented by $\tilde{T} := T \cup U$, where $T = \{(x_1, y_1), \ ... \ , (x_m, y_m)\}$ represents the labeled training data, with data points $x_i \in \mathbb{R}^m$ and labels $y_i \in \{-1, 1\}$ for $i = 1, ..., m$, and $U = \{x_1^*, \ ... \ , x_u^*\}$ represents the Universum data. $\|.\|$ denotes the Euclidean norm, $\otimes$ denotes the Kronecker product, a vector of arbitrary dimension is represented by e, and I represents the identity matrix. Additionally, all vectors in this paper are column vectors, denoted by a superscript T for transposition, representing row vectors. Some important nations in this paper are shown in Table 2.

Table 2. List of notations.

Notations	Description	Notations	Description
$:=$	means "define".	X_i	$:=I_n \otimes x_i^T$.
M_i	$:=X_i D_n$.	H_i	$:=[M_i \ I_n]$.
z	$:=[hvec(W) \ b]$.	G	$:=2\sum_{i=1}^m H_i^T H_i$.
s_i	$:=\frac{1}{2}hvec(x_i x_i^T)$.	s_j	$:=\frac{1}{2}hvec(u_j u_j^T)$.
r_i	$:=[s_i \ x_i]$.	r_j	$:=[s_j \ u_j]$.

Furthermore, we have introduced the symmetric matrix vectorization technique, which is widely used in the reconstruction of kernel-free SVM models [35, 38]. Considering a square matrix $W := (W_{ij}) \in \mathbb{R}^{n\times n}$, its vectorization can be represented as:

$$vec(W) := [W_{11}, \cdots, W_{n1}, W_{12}, \cdots, W_{n2}, \cdots, W_{1n}, \cdots, W_{nn}]^T \in \mathbb{R}^{n\times n}.$$

If $W \in \mathbb{S}^n$ is a symmetric matrix, then its half-vectorization can be represented as:

$$hvec(W) := [W_{11}, \cdots, W_{1n}, W_{22}, \cdots, W_{2n}, \cdots, W_{n-1,n-1}, W_{n-1,n}, W_{nn}]^T \in \mathbb{R}^{\frac{n\times(n+1)}{2}}.$$

Given $n \in \mathbb{N}$, there exists a unique elimination matrix $L_n \in \mathbb{R}^{\frac{n\times(n+1)}{2}\times n^2}$, and a unique duplication matrix $D_n \in \mathbb{R}^{n^2\times\frac{n\times(n+1)}{2}}$, such that the following equation holds [39]:

$$L_n vec(W) = hvec(W), \ D_n hvec(W) = vec(W), \forall \ W \in S_n.$$

If $W \in \mathbb{D}^n$ is a diagonal matrix, the diag-vectorization of W is defined as follows:

$$dvec(W) := [W_{11}, W_{22}, \cdots, W_{n-1,n-1}, W_{nn}]^T \in \mathbb{R}^n.$$

For any vector $a = [a_1, \cdots, a_n]$, the quadratic vector $qvec(a)$ is defined as follows:

$$qvec(a) := [\frac{1}{2}a_1^2, \frac{1}{2}a_2^2, \cdots, \frac{1}{2}a_n^2]^T \in \mathbb{R}^n,$$

and the quadratic vector $lvec(a)$ with cross terms is as follows:

$$lvec(a) := [\frac{1}{2}a_1^2, a_1a_2, \cdots, a_1a_n, \frac{1}{2}a_2^2, a_2a_3, \cdots, a_2a_n, \cdots, \frac{1}{2}a_n^2]^T \in \mathbb{R}^{\frac{n\times(n+1)}{2}}.$$

Remark. For $\forall\ i = 1, \cdots, m;\ \ \forall\ j = 1, \cdots, u$, in Table 2, and the construction of M^i is as follows: Let M^i_{jp} be the pth column and jth row element of M^i, if $hvec(W)$ is w_{jp} or w_{pj}, then $M^i_{jp} = x^k_i$. Otherwise, we assign it to be 0.

According to the above definition, we can obtain the following conclusions:

$$\begin{aligned}
&Wx_i = I_n \otimes x_i^T vec(W) = X_i vec(W) = X_i D_n hvec(W) = M_i hvec(W),\\
&Wx_i + b = M_i hvec(W) + I_n b = H_i z,\\
&\sum_{i=1}^{m} \|Wx_i + b\|_2^2 = \sum_{i=1}^{m} (H_i z)^T H_i z = z^T [\sum_{i=1}^{m} H_i^T H_i] z = z^T G z,\\
&\frac{1}{2} x_i^T W x_i = lvec(x_i)^T hvec(W),\\
&\frac{1}{2} x_i^T W x_i + b^T x_i = [hvec(W)\ b]^T [\frac{1}{2} hvec(x_i x_i^T)\ x_i] = [hvec(W)\ b]^T [s_i\ x_i] = z^T r_i.
\end{aligned} \tag{2.1}$$

2.2. Universum support vector machine

U-SVM adds a Universum term to the standard SVM, and its primal mathematical formula is:

$$\min \frac{1}{2}\|\omega\|_2^2 + C \sum_{i=1}^{m} H_\varepsilon[y_i f_{w,b}(x_i)] + C_u \sum_{i=1}^{u} U[f_{w,b}(x_i^*)], \tag{2.2}$$

where $f_{w,b}$ represents a classifier, $H_\varepsilon[t] = \{\max 0, -\varepsilon - t\}$ represents the hinge loss, and $U[t] = U_{-\varepsilon}[t] + U_{-\varepsilon}[-t]$ represents the ε-insensitive loss (as shown in Figure 1(a)). Prior knowledge in U-SVM is encompassed in $\sum_{i=1}^{u} U[f_{w,b}(x_i^*)]$; the smaller this value, the more prior knowledge is incorporated into $f_{w,b}$, and vice versa [24]. For the binary classification problem, the Universum data does not belong to the positive class or the negative class; instead, it must fall between the hyperplanes of the two classes. To achieve this, separate hyperplanes with large amounts of Universum data (dashed lines) should be selected (as shown in Figure 1(b)) [40]. In addition, the prior knowledge embedded in the Universum cannot be directly represented by $U[t]$ [41]. Therefore, formula (2.2) can be further expressed as:

$$\begin{aligned}
&\min_{\omega,b,\varepsilon,\xi_j} \frac{1}{2}\|\omega\|_2^2 + C \sum_{i=1}^{m} \xi_i + C_u \sum_{j=1}^{u} \xi_j\\
&s.t.\ \ y_i[(\omega x_i) + b] \geq 1 - \xi_i, \xi_i \geq 0, i = 1, ..., m,\\
&\qquad y_j[(\omega x_j) + b] \geq -\varepsilon - \xi_j, \xi_j \geq 0, j = 1, ..., u,
\end{aligned} \tag{2.3}$$

where the first two terms correspond to the standard soft-margin SVM, while the last term pertains to Universum data in the objective function. ξ_i and ξ_j represent the slack variables in standard soft-margin SVM and Universum data. $C,\ C_u \geq\ 0$ control the trade-off between minimizing training errors and maximizing the number of Universum data, respectively. When $C_u = 0$, formula (2.3) is reduced to a standard SVM classifier [42].

2.3. Quadratic surface support vector machine

To avoid using kernel functions for nonlinear classification of data, QSSVM achieves this goal by utilizing the quadratic surface: $f(x) = \frac{1}{2}x^T Wx + b^T x + c = 0$, where $W \in \mathbb{S}^{m\times m}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}$. It separates the training points into two classes by maximizing the distance from each data point to the separating surface. The QSSVM model is given as follows:

$$\begin{aligned} &\min_{W,b,c} \sum_{i=1}^{m} \|Wx_i + b\|_2^2 \\ &\text{s.t. } y_i(\frac{1}{2}x_i^T Wx_i + b^T x_i + c) \geq 1, \ \ i = 1, ..., m. \end{aligned} \tag{2.4}$$

In formula (2.4), consider the presence of noise and outliers that may exist in the data. Luo et al. [32] proposed the SQSSVM, which penalizes misclassifications. The SQSSVM model is given as follows:

SQSSVM:

$$\begin{aligned} &\min_{W,b,c} \sum_{i=1}^{m} \|Wx_i + b\|_2^2 + C\sum_{i=1}^{m} \xi_i \\ &\text{s.t. } y_i(\frac{1}{2}x_i^T Wx_i + b^T x_i + c) \geq 1 - \xi_i, \ \xi_i \geq 0, \ i = 1, ..., m, \end{aligned} \tag{2.5}$$

where $C > 0$ is the penalty parameter and ξ_i represents the slack variable. According to formula (2.1), we can derive the dual form of SQSSVM (SQSSVM′):

SQSSVM′:

$$\begin{aligned} &\min_{z,c,\xi} \ z^T Gz + C_l \sum_{i=1}^{m} \xi_i \\ &\text{s.t. } y_i(s_i^T z + c) \geq 1 - \xi_i, \ \ i = 1, ..., m, \\ &\qquad \xi_i \geq 0, \ \ i = 1, ..., m. \end{aligned} \tag{2.6}$$

By solving problem (2.6), z and c can be calculated, and then W and b can be determined. A new data point $x \in \mathbb{R}^m$ can be assigned to a class by calculating

$$f(x) = sign\,(\frac{1}{2}x^T Wx + b^T x + c),$$

where f(x) represents the decision function.

3. Main contribution

3.1. Motivation

In the above discussion, we mentioned that SQSSVM can handle highly nonlinear separable datasets more efficiently than other SVMs because it avoids the need to select kernel functions. Meanwhile, Universum provides additional prior knowledge that can enhance generalization performance and prevent overfitting. However, SQSSVM's utilization of the L_2-norm distance makes it sensitive to outliers, particularly in the presence of heavy noise. Moreover, the abundance of slack variables can

also impact classification performance. Furthermore, Universum's performance in robust learning is not ideal.

To address these challenges, we propose a novel framework called A-R-U-SQSSVM. The overall architecture is shown in Figure 2. To begin, we improve SQSSVM by replacing the L_2-norm with a capped $L_{2,p}$-norm distance metric for better robustness and substituting the misclassification term with a generalized Welsch loss function for improved adaptability. We also combine the improved SQSSVM with Universum data to further enhance its generalization performance. Additionally, we utilize R-U-SQSSVM as a weak classifier and the AdaBoost algorithm to enhance classification performance, culminating in our strong classifier, the A-R-U-SQSSVM.

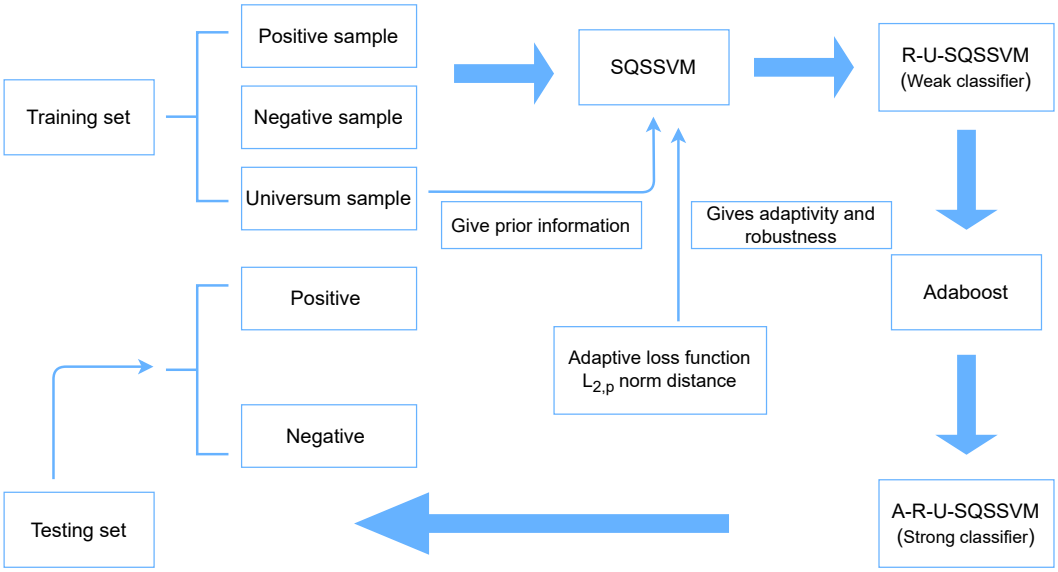


Figure 2. Overall architecture of the proposed A-R-U-SQSSVM.

3.2. Proposed R-U-SQSSVM model

Therefore, the optimization problem of RUSQSSVM with Universum data constraints can be expressed as follows:

$$\begin{aligned} &\min_{W,b,c,\xi_i,\xi_j^*} \sum_{i=1}^{m} \min(\|Wx_i + b\|_2^p, \varepsilon) + C \sum_{i=1}^{m} \frac{\sigma^2}{2}[1 - \exp(-\frac{\xi_i^2}{2\sigma^2})]^\theta + C_u \sum_{j=1}^{u} \xi_j^* \\ &s.t.\ \ y_i(\frac{1}{2}x_i^T W x_i + b^T x_i + c) \geq 1 - \xi_i\,, \quad \xi_i \geq 0, \ \ i = 1, \cdots, m, \\ &\qquad y_j(\frac{1}{2}u_j^T W u_j + b^T u_j + c) \geq -\varepsilon - \xi_j^*\,, \xi_j^* \geq 0, \ \ j = 1, \cdots, u. \end{aligned} \tag{3.1}$$

• The parameters C and C_u are penalty parameters that trade off the misclassification of SQSSVM and the margin of Universum data, respectively.

• The parameters ξ_i $(i = 1, ..., m)$ and ξ_j^* $(j = 1, ..., u)$ represent the slack variables of the training sample and the Universum sample.

• The item $\sum_{i=1}^{m} \min(\|Wx_i + b\|_2^p, \varepsilon)$ represents the improved loss function. Here, $\varepsilon > 0$ is a threshold parameter, and $0 < p \leq 2$. By utilizing $\|.\|_2^p$, we ensure that the value of the misclassification rate is at most ε.

• The item $C\sum_{i=1}^{m_2}\frac{\sigma^2}{2}[1-\exp(-\frac{\xi_i^2}{2\sigma^2})]^\theta$ represents the improved Welsch loss. Here, σ represents the penalty parameter, and $\theta > 0$ is an adaptive parameter of the loss function.

• The item $C_u\sum_{j=1}^{m_u}\xi_j^*$ represents the Universum item.

3.2.1. Optimization problems to solve R-U-SQSSVM

As shown in formula (3.1), R-U-SQSSVM is neither convex nor concave. Therefore, it will be solved in the following three steps. The first step uses the HQ algorithm to optimize the loss function term, which is the second term of formula (3.1); the second step uses the concave duality theorem to optimize the distance metric term, which is the first term of formula (3.1); the third step uses the Lagrangian function and Karush-Kuhn-Tucker (KKT) conditions to transform the primal problem into its dual form.

First, we use the HQ algorithm [43] and the theory of conjugate functions to solve the second term:

$$\min C\sum_{i=1}^{m}\frac{\sigma^2}{2}[1-\exp(-\frac{\xi_i^2}{2\sigma^2})]^\theta. \tag{3.2}$$

We can write formula (3.2) as:

$$\max_{\alpha} G_1(\alpha)$$

where

$$G_1(\alpha) = C\sum_{i=1}^{m}\frac{\sigma^2}{2}[\exp(-\frac{\xi_i^2}{2\sigma^2})]^\theta.$$

To facilitate the following derivations, we define the convex function $g_\theta(v)$ as:

$$g_\theta(v) = \frac{1}{\theta}[-v\log(-v)+v]^\theta, \quad v<0.$$

From the theory of conjugate functions, we obtain:

$$\exp(-\frac{\xi_i^2}{2\sigma^2}) = \sup_{v<0}[v\frac{\xi_i^2}{2\sigma^2} - g_\theta(v)], \quad v = -\exp(-\frac{\xi_i^2}{2\sigma^2}).$$

Thus, we can obtain:

$$\max_{\alpha,v<0}\{\sum_{i=1}^{m}[v\frac{\xi_i^2}{2\sigma^2} - g_\theta(v)]\}$$

which is equivalent to:

$$\max_{\alpha}\{\sum_{i=1}^{m}[-\exp(-\frac{\xi_i^2}{2\sigma^2})]\frac{\xi_i^2}{2\sigma^2} - g_\theta[-\exp(-\frac{\xi_i^2}{2\sigma^2})]\}.$$

This can be optimized by utilizing the HQ optimization algorithm, where we alternate between optimizing α and v. Specifically, given a fixed $\alpha^{(s)}$, we can solve for $v_i^{(s)}$ using the same equation as before.

$$v_i^{(s)} = -\exp(-\frac{(\xi_i^{(s)})^2}{2\sigma^2})^{\theta}.$$

Therefore, the second term can be rewritten as:

$$\min C\sum_{i=1}^{m}\frac{\sigma^2}{2}[1-\exp(-\frac{\xi_i^2}{2\sigma^2})]^{\theta} = \frac{C}{2\sigma^2}\kappa^T\Omega^{\theta}\kappa \tag{3.3}$$

where $\kappa = [\xi_1, \xi_2, \cdots, \xi_m] \in \mathbb{R}^m$, $\Omega = [-\theta\exp(-\frac{\xi_1^2}{2\sigma^2})^{\theta}, -\theta\exp(-\frac{\xi_2^2}{2\sigma^2})^{\theta}, \cdots, -\theta\exp(-\frac{\xi_m^2}{2\sigma^2})^{\theta}] \in \mathbb{D}^{m\times m}$.

Based on the formulas (3.1) and (3.3), the formula (3.1) can be rewritten as:

$$\begin{aligned}&\min_{W,b,c,\xi_i,\xi_j^*}\sum_{i=1}^{m}\min(\|Wx_i+b\|_2^p,\varepsilon)+\frac{C}{2\sigma^2}\kappa^T\Omega^{\theta}\kappa+C_u\sum_{j=1}^{u}\xi_j^*\\ &s.t.\ \ y_i(s_i^Tz+c)\geq 1-\xi_i\ ,\quad \xi_i\geq 0,\ \ i=1,\cdots,m,\\ &\qquad y_j(s_j^Tz+c)\geq -\varepsilon-\xi_j^*\ ,\xi_j^*\geq 0,\ \ j=1,\cdots,u.\end{aligned} \tag{3.4}$$

At this point, the first step is complete. Then, we begin the second step:

Next, we use the concave duality theorem [43, 44] to solve the first term in formula (3.1):

$$\min_{W,b}\sum_{i=1}^{m}\min(\|Wx_i+b\|_2^p,\varepsilon). \tag{3.5}$$

Lemma 1. *Let $g(\theta) : \mathbb{R}^n \to \mathbb{R}$ be a continuous non-convex function, and suppose $h(\theta) : \mathbb{R}^n \to \Xi$ is a mapping with a range in the convex hull Ξ. Assume there exists a concave function $\bar{g}(\theta)$ defined on Ξ such that $g(\theta) = \bar{g}(h(\theta))$. Therefore, the non-convex function $g(\theta)$can be expressed as:*

$$g(\theta) = \inf_{v\in\mathbb{R}^n}[v^Th(\theta) - g^{*(v)}]. \tag{3.6}$$

According to the concave duality theorem, $g^*(v)$ is the concave dual of $\bar{g}(u)$, and its expression is as follows:

$$g^*(v) = \inf_{u\in\Xi}[v^Tu - \bar{g}(u)].$$

In addition, the minimum value on the right side is as follows:

$$v^* = \frac{\partial\bar{g}(u)}{\partial u}|_{u=h(\theta)}.$$

Based on Lemma 1, define a concave function $\bar{g}(\theta) : \mathbb{R} \to \mathbb{R}$. For any $z > 0$, we have

$$\bar{g}(\theta) = \min(\theta^{\frac{p}{2}}, \varepsilon).$$

Assuming $h(U) = U^2$, we can obtain:

$$\min(\|Wx_i + b\|_2^p, \varepsilon) = \bar{g}(h(U)) \ , U = \|Wx_i + b\|_2^2.$$

Therefore, based on formula (3.7), the first term of formula (3.1) can be rewritten as:

$$\min \sum_{i=1}^{m} [\sum_{i=1}^{m} \min(\|Wx_i + b\|_2^p, \varepsilon)] = \min \sum_{i=1}^{m} \bar{g}(\|Wx_i + b\|_2^2). \tag{3.7}$$

By utilizing formula (3.6), the terms in formula (3.7) can be further expressed as:

$$\min(\|Wx_i + b\|_2^p, \varepsilon) = \bar{g}(\|Wx_i + b\|_2^2) = \inf_{d_{ii} \geq 0} (d_{ii} h(U) - g^*(d_{ii})) = \inf_{d_{ii} \geq 0} d_{ii}\theta - g^*(d_{ii}). \tag{3.8}$$

Thus, the concave dual function of the given $\bar{g}(\theta)$ is:

$$g^*(d_{ii}) = \inf_{\theta} [d_{ii} z - \bar{g}(z)] = \inf_{z} \begin{cases} d_{ii}\theta - \theta^{\frac{p}{2}} \ , \theta^{\frac{p}{2}} < \varepsilon, \\ d_{ii}\theta - \varepsilon_1 \ \ , \theta^{\frac{p}{2}} \geq \varepsilon. \end{cases} \tag{3.9}$$

By optimizing z in formula (3.9), we can obtain:

$$g^*(d_{ii}) = \begin{cases} d_{ii}(\frac{2}{p} d_{ii})^{\frac{2}{p-2}} - (\frac{2}{p} d_{ii})^{\frac{2}{p-2}}, \ \theta^{\frac{p}{2}} < \varepsilon, \\ d_{ii}\varepsilon^{\frac{2}{p}} - \varepsilon, \qquad\qquad\qquad \theta^{\frac{p}{2}} \geq \varepsilon. \end{cases} \tag{3.10}$$

Therefore, the first term in the objective function (3.1) can be further simplified as:

$$\min(\|Wx_i + b\|_2^p, \varepsilon) = \inf_{d_{ii} \geq 0} L_i(W, b, d_{ii}, \varepsilon),$$

where

$$L_i(W, b, d_{ii}, \varepsilon) = \begin{cases} d_{ii}\theta - d_{ii}(\frac{2}{p} d_{ii})^{\frac{2}{p-2}} + (\frac{2}{p} d_{ii})^{\frac{2}{p-2}}, \qquad \theta^{\frac{p}{2}} < \varepsilon, \\ d_{ii}\theta - d_{ii}\varepsilon^{\frac{2}{p}} + \varepsilon, \qquad\qquad\qquad\qquad \theta^{\frac{p}{2}} \geq \varepsilon. \end{cases}$$

Therefore, the first term in formula (3.1) can be rewritten as:

$$\min_{W,b} \sum_{i=1}^{m} \min(\|Wx_i + b\|_2^p, \varepsilon) \Leftrightarrow \min_{W,b} \sum_{i=1}^{m} \inf_{d_{ii} \geq 0} L_i(W, b, d_{ii}, \varepsilon) \Leftrightarrow \min_{W,b,d_{ii} \geq 0} \sum_{i=1}^{m} L_i(W, b, d_{ii}, \varepsilon). \tag{3.11}$$

The optimal classifier is learned through an alternating optimization algorithm to solve the objective function (3.11). The gradient of the function $\bar{g}(\theta)$ with respect to θ is expressed as:

$$\frac{\partial \bar{g}(\theta)}{\partial z} = \begin{cases} \frac{p}{2}\theta^{\frac{p}{2}-1}, \qquad 0 < \theta < \varepsilon^{\frac{2}{p}}, \\ 0, \qquad\qquad\quad \theta > \varepsilon^{\frac{2}{p}}. \end{cases}$$

If $\theta = h(U) = \|Wx_i + b\|_2^2$, by fixing W and b, we can obtain:

$$d_{ii} = \frac{\partial \bar{g}(\theta)}{\partial \theta}|_{\theta=\|Wx_i+b\|_2^2} = \begin{cases} \frac{p}{2}\|Wx_i + b\|_2^{p-2}, & 0 < \|Wx_i + b\|_2^p < \varepsilon, \\ 0, & else. \end{cases} \tag{3.12}$$

To better understand the relationship between the parameters, this paper defines the distance from the sample x_i to the hyperplane as X. If $X > \varepsilon$ and d_{ii} is almost equal to 0, then the sample x_i is considered an outlier and is discarded. Additionally, when the variable d_{ii} is fixed, we aim to solve for the parameters W and b related to the classifier. Thus, the objective function of the original optimization problem (3.1) can be written as:

$$\min_{W,b,c} \sum_{i=1}^{m} \min_{W,b} \sum_{i=1}^{m} d_{ii}(\|Wx_i + b\|_2^2) + \frac{C}{2\sigma^2}\kappa^T \Omega^\theta \kappa + C_u \sum_{j=1}^{u} \xi_j^*. \tag{3.13}$$

Let $D = (d_{11}, \cdots, d_{m,m})$ be an $m \times m$ diagonal matrix. The above optimization problem (3.13) can be rewritten as:

$$\begin{aligned} &\min_{W,b,c,\xi_i,\xi_j^*} \sum_{i=1}^{m} D\|Wx_i + b\|_2^2 + \frac{C}{2\sigma^2}\xi^T \Omega^\theta \xi + C_u \sum_{j=1}^{u} \xi_j^* \\ &s.t.\ \ y_i(\frac{1}{2}x_i^T W x_i + b^T x_i + c) \geq 1 - \xi_i\,, \quad \xi_i \geq 0, \ \ i = 1, \cdots, m, \\ &\quad y_j(\frac{1}{2}u_j^T W u_j + b^T x_j + c) \geq -\varepsilon - \xi_j^*\,, \xi_j^* \geq 0, \ \ j = 1, \cdots, u. \end{aligned} \tag{3.14}$$

The corresponding Lagrangian function for formula (3.14) can be written as:

$$\begin{aligned} L(W, b, c, \xi, \xi^*, \alpha, \beta, \gamma, \eta) = &\sum_{i=1}^{m} D\|Wx_i + b\|_2^2 + \frac{C}{2\sigma^2}\xi^T \Omega^\theta \xi + C_u \sum_{j=1}^{u} \xi_j^* \\ &+ \sum_{i=1}^{m} \alpha_i \left[1 - \xi_i - y_i\left(\frac{1}{2}x_i^T W x_i + b^T x_i + c\right)\right] \\ &+ \sum_{j=1}^{u} \beta_j \left[-\varepsilon - \xi_j^* - y_j\left(\frac{1}{2}u_j^T W u_j + b^T u_j + c\right)\right] \\ &+ \sum_{i=1}^{m} \gamma_i \xi_i + \sum_{j=1}^{u} \eta_j \xi_j^* \end{aligned} \tag{3.15}$$

where $\alpha_i \geq 0, \beta_j \geq 0, \gamma_i \geq 0, \eta_j \geq 0$ are Lagrange multipliers, α_i corresponds to the training sample constraints, β_j corresponds to the Universum constraints, and γ_i and η_j correspond to the slack variables. By taking derivatives, we can obtain the complete KKT conditions:

$$\begin{cases} \dfrac{\partial L}{\partial W} = 2D\sum_{i=1}^{m}(Wx_ix_i^T + bx_i^T) - \sum_{i=1}^{m}\alpha_i y_i x_i x_i^T - \sum_{j=1}^{u}\beta_j y_j u_j u_j^T = 0. & (1) \\ \dfrac{\partial L}{\partial b} = 2D\sum_{i=1}^{m}(Wx_i + b) - \sum_{i=1}^{m}\alpha_i y_i x_i - \sum_{j=1}^{u}\beta_j y_j u_j = 0. & (2) \\ \dfrac{\partial L}{\partial c} = -\sum_{i=1}^{m}\alpha_i y_i - \sum_{j=1}^{u}\beta_j y_j = 0. & (3) \\ \dfrac{\partial L}{\partial \xi} = \dfrac{C}{\sigma^2}\Omega^{\{\theta\}}\xi - \alpha_i - \gamma_i = 0 \Rightarrow \xi = \dfrac{\sigma^2}{C}\Omega^{-\{\theta\}}(\alpha_i + \gamma_i). & (4) \\ \dfrac{\partial L}{\partial \xi_j^*} = C_u - \beta_j - \eta_j = 0 \Rightarrow \beta_j = C_u - \eta_j. & (5) \\ \alpha_i[1 - \xi_i - y_i(\dfrac{1}{2}x_i^T W x_i + b^T x_i + c)] = 0, \gamma_i\xi_i = 0. & (6) \\ \beta_j[-\varepsilon - \xi_j^* - y_j(\dfrac{1}{2}u_j^T W u_j + b^T u_j + c)] = 0, \eta_j\xi_j^* = 0. & (7) \end{cases} \tag{3.16}$$

To smoothly derive the quadratic programming form of the dual problem from the primal problem, we first express the redundant primal variables W, b, c, ξ, ξ^* in terms of dual variables:

Step one: Eliminate W and b. From the first two steps in the formula , we can solve for the expressions of W and b. Define $z = [\text{hvec}(W)\ b]$, and using the notation $H_i = [M_i\ I_n]$ from Table 2, we have:

$$D\sum_{i=1}^{m}H_i^T H_i z = \sum_{i=1}^{m}\alpha_i y_i r_i + \sum_{j=1}^{u}\beta_j y_j r_j,$$

where $r_i = [s_i\ x_i]$ and $r_j = [s_j\ u_j]$. Furthermore, z can be expressed as:

$$z = (D\sum_{i=1}^{m}H_i^T H_i)^{-1}(\sum_{i=1}^{m}\alpha_i y_i r_i + \sum_{j=1}^{u}\beta_j y_j r_j).$$

Step two: Eliminate ξ_i and ξ_j^*. From $\xi_i = \frac{\sigma^2}{C}\Omega^{-\theta}(\alpha_i + \gamma_i)$ and $\beta_j = C_u - \eta_j$, combined with the complementary slackness conditions $\gamma_i\xi_i = 0$ and $\eta_j\xi_j^* = 0$, we can deduce that $\gamma_i = 0$ and $\eta_j = 0$ (when $\xi_i > 0$ or $\xi_j^* > 0$).

Then, substituting the above expressions into the Lagrangian function, we obtain the dual problem:

$$\min_{\alpha,\beta}\frac{1}{2}(\sum_{i=1}^{m}\alpha_i y_i r_i + \sum_{j=1}^{u}\beta_j y_j r_j)^T(D\sum_{i=1}^{m}H_i^T H_i)^{-1})(\sum_{i=1}^{m}\alpha_i y_i r_i + \sum_{j=1}^{u}\beta_j y_j r_j) - \sum_{i=1}^{m}\alpha_i + \varepsilon\sum_{j=1}^{u}\beta_j. \tag{3.17}$$

Define the vector $\lambda = [\alpha\ \beta]$ and the matrix:

$$Q = \begin{bmatrix} \sum_{i=1}^{m} y_i r_i r_i^T & \sum_{i,j} y_i y_j r_i r_j^T \\ \sum_{i,j} y_i y_j r_j r_i^T & \sum_{j=1}^{u} y_j r_j r_j^T \end{bmatrix}, \quad P = D\sum_{i=1}^{m}H_i^T H_i. \tag{3.18}$$

According to the above definition, the dual problem can be further expressed as:

$$\begin{aligned} &\min_{\lambda} \frac{1}{2}\lambda^{\top} Q P^{-1} Q^{\top} \lambda - \lambda^{\top} \begin{bmatrix} e \\ -\varepsilon \end{bmatrix}, \\ &s.t. \sum_{i=1}^{m} \alpha_i y_i + \sum_{j=1}^{u} \beta_j y_j = 0,\ 0 \le \alpha_i \le \frac{C}{\sigma^2}\Omega^{\theta},\ 0 \le \beta_j \le C_u, \end{aligned} \tag{3.19}$$

where e is a vector with all elements equal to 1, suitable for the dimension.

Therefore, it can be seen that the formula (3.19) is a convex optimization problem. By solving it, we can calculate z and c. Subsequently, the optimal parameter set for the quadratic surface (W, b, c) can be determined. By calculation, new data points $x \in \mathbb{R}^m$ can be assigned to the corresponding categories.

$$f(x) = sign\,(\frac{1}{2}x^T W x + b^T x + c).$$

The pseudo-code for U-R-SQSSVM is presented in Algorithm 1.

Algorithm 1 R-U-SQSSVM algorithm.

Input: Training set $A \in \mathbb{R}^{m\times n}$ and Universum set $U \in \mathbb{R}^{u\times n}$; parameters C, C_u, σ, ε, θ, p; max iterations $T_{\max}$; convergence threshold δ.

Output: Optimal classifier parameters (W^*, b^*, c^*).

1. Initialize diagonal matrices $D = \mathrm{diag}(d_{11}, \ldots, d_{mm})$ and $\Omega = \mathrm{diag}(-\theta \exp(-\xi_i^2/(2\sigma^2))^{\theta})$.

Initialize $W^{(0)}, b^{(0)}, c^{(0)}$ to zero vectors.

Compute initial objective value $O^{(0)}$.

2. Iteratively update parameters:

for $t = 1$ to $T_{\max}$ **do**

Update $D^{(t)}$ via (3.12).

Solve dual problem (3.19) for α, β.

Compute primal variables:

$$W = (D\sum H_i^T H_i)^{-1}(\sum \alpha_i y_i x_i x_i^T + \sum \beta_j y_j u_j u_j^T),$$

$$b = (D\sum H_i^T H_i)^{-1}(\sum \alpha_i y_i x_i + \sum \beta_j y_j u_j).$$

Compute $\xi_i = \frac{\sigma^2}{C\Omega^p}(\alpha_i + \gamma_i)$.

if $|O^{(t)} - O^{(t-1)}| < \delta$ **then**

Break loop.

end if

end for

3. Return optimal parameters (W^*, b^*, c^*).

3.3. AdaBoost algorithm based on R-U-SQSSVM

The boosting algorithm is a popular ensemble learning technique. Its principle is to iteratively train a weak classifier and combine the results of different weak classifiers to form a final strong classifier.

Classical examples of boosting algorithms include Boosting [46], Adaptive Boosting (AdaBoost) [47], gradient boosting decision tree [48], extreme gradient Boosting (XGBoost) [49], etc. In binary classification problems, the primary concept of AdaBoost is to determine the classification outcome by calculating the weights of each weak classifier. During the training process, each weak classifier is assigned a weight based on its classification error rate in the previous iteration. If a sample is misclassified, its weight will increase; otherwise, it will decrease. In recent years, reference [50] has proposed a boosting algorithm called U-AdaBoost, which can improve AdaBoost's classification performance for Universum data. Reference [51] designed a boosting-based Universum algorithm for semi-supervised learning. They also demonstrate that the training error of AdaBoost with Universum is bounded by the product of the normalization factor, and the training error decreases exponentially quickly when each weak classifier is slightly better than random guessing. Reference [52] first combined Universum data with an improved twin support vector machine (TSVM) to enhance its generalization performance. They then integrated the AdaBoost method into their proposed model to further enhance the learning effectiveness.

Based on the above discussion, our A-R-U-SQSSVM framework can be formulated as:

$$\begin{aligned}
&\min_{W,b,c,\xi_i,\xi_j^*} \sum_{i=1}^{m} \mu_i^{(t)} [\sum_{i=1}^{m} \min(\|Wx_i + b\|_2^p, \varepsilon) + C \sum_{i=1}^{m} \frac{\sigma^2}{2} [1 - \exp(-\frac{\xi_i^2}{2\sigma^2})]^\theta] + C_u \sum_{j=1}^{u} \xi_j^* \\
&s.t.\ \ y_i(\frac{1}{2} x_i^T W x_i + b^T x_i + c) \geq 1 - \xi_i\,, \quad \xi_i \geq 0,\ \ i = 1, ..., m, \\
&\qquad y_j(\frac{1}{2} u_j^T W u_j + b^T x_j + c) \geq -\varepsilon - \xi_j^*\,, \xi_j^* \geq 0,\ \ j = 1, ..., u.
\end{aligned} \tag{3.20}$$

where μ is the weighting function utilized in the AdaBoost algorithm, and the weights $\mu_i^{(t)}$ are employed to reweigh the misclassified samples in the objective function of A-R-U-SQSSVM.

Specifically, we utilize R-U-SQSSVM as the weak classifier and obtain a strong classifier, A-R-U-SQSSVM, by employing AdaBoost methods, which yields a more robust classification effect for our subsequent experiments. For details, please refer to Algorithm 2.

Algorithm 2 A-R-U-SQSSVM algorithm.

Input: Training set $A \in \mathbb{R}^{m\times n}$ and Universum set $U \in \mathbb{R}^{u\times n}$; parameters C, C_u, σ, ε, θ, p; max iterations $T_{\max}$; boosting rounds K.

Output: Strong classifier $H(x) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k h_k(x)\right)$.

1. Initialize sample weights $\mu_i^{(1)} = \frac{1}{m}$ for $i = 1, \ldots, m$.

2. Ensemble weak classifiers:

for $k = 1$ to K **do**

Train RUSQSSVM $h_k(x)$ with weights $\mu^{(k)}$.

Compute error $\varepsilon_k = \sum_{i=1}^{m} \mu_i^{(k)} I(y_i \neq h_k(x_i)) + \sum_{j=1}^{u} I(0 \neq h_k(u_j))$

if $\varepsilon_k > 0.5$ **then**

Re-initialize weights

continue

end if

Set $\alpha_k = 0.5 \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$

Update weights:

$$\mu_i^{(k+1)} = \frac{\mu_i^{(k)} \exp(-\alpha_k y_i h_k(x_i))}{Z_k} \quad \forall i = 1, \ldots, m$$

end for

3. Aggregate classifiers:

return $H(x) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k h_k(x)\right)$

4. Numerical experiments

In this section, the paper further explores the effectiveness, robustness, and classification performance of the proposed A-R-U-SQSSVM model through numerical experiments. This chapter first introduces the experimental setup, then presents the experimental results on the UCI dataset, and analyzes the impact of the Universum data ratio on the classification results. Finally, this chapter compares the performance differences between the proposed method and other algorithms through statistical analysis.

4.1. Experimental setup

The A-R-U-SQSSVM is compared with the following algorithms: The CTSVM, SQSSVM [32], random forest (RF), kernel-free fuzzy support vector machine with Universum (KFFUSVM), XGBoost, ensemble pruning based on classification confidence (EPCC) [54], weight-constrained neural networks (WCNN_1) [55], and adaptive boosting algorithm based on support vector machine (ASVM) [55]. All the aforementioned algorithms are implemented in MATLAB R2021, and the experimental setup consists of a personal computer equipped with a 2.11 GHz CPU and 16GB of memory.

Twelve datasets were selected from the UCI benchmark database*, including Balance, Australian, Hepat, German, Ionosp, QSAR, Sonar, a8a, a7a, a5a, a3a, and a1a. For specific information, see

*https://archive.ics.uci.edu/ml/index.php/.

Table 3. In Table 3, "n" represents the dimensions of each dataset, while "m_1" and "m_2" represent the number of positive and negative samples in each dataset, respectively. First, the downloaded raw data is normalized to keep the data within the range of [0,1], in order to reduce the differences between the features of different samples. Then, each dataset is randomly divided into training and testing sets, with a sample ratio of 7:3. Finally, to reduce the impact of random experimental results, 10-fold cross-validation is performed on each dataset. In addition, to verify the robustness of the proposed model, experiments were conducted in environments with Gaussian noise of $N(0, 0^2)$, $N(0, 0.15^2)$, and $N(0, 0.3^2)$. The Gaussian noise follows a normal distribution $N(\mu, \sigma^2)$, where μ represents the mean of the Gaussian noise distribution and σ^2 represents the variance of the Gaussian noise distribution.

Table 3. Description of the UCI datasets.

Datasets	n	m_1 vs. m_2	Datasets	n	m_1 vs. m_2
Balance	4	288 vs. 288	Sonar	60	111 vs. 97
Australian	14	307 vs. 383	a8a	123	6906 vs. 2959
Hepat	19	123 vs. 32	a7a	123	11523 vs. 4938
German	24	700 vs. 300	a5a	123	18303 vs. 7844
Ionosp	34	225 vs. 125	a3a	123	20564 vs. 8812
QSAR	41	699 vs. 356	a1a	123	21670 vs. 9286

• Evaluation Criteria:

To fairly evaluate the classification performance of the nine models mentioned above, we measured their performance using the classical metrics of accuracy (ACC) and F_1-score (F_1) [46]. The formulas for calculating ACC and F_1 are as follows:

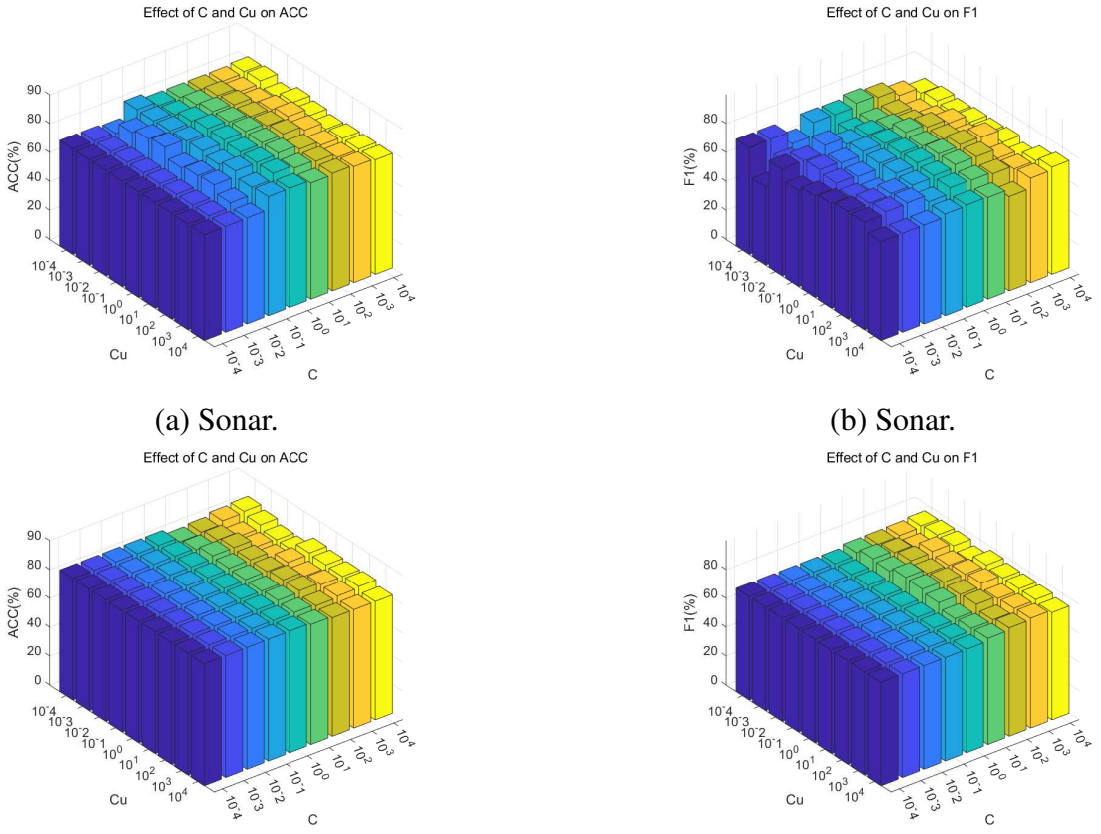
$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \quad F_1 = \frac{2TP}{2TP + FN + FP}, \tag{4.1}$$

where TP and TN represent true positives and true negatives, respectively, while FP and FN represent false positives and false negatives, respectively. Higher values of ACC and F_1 indicate better performance. In addition, the ACC and F_1 values we recorded represent the average results of nine algorithms running ten times on ten datasets, and the recorded time (s) refers to the running time of each algorithm on different datasets. The final experimental results of the above algorithms are derived from the test sets of various datasets, with the best ACC and F_1 values highlighted in bold.

• Parameter Selection:

The penalty parameter values for the nine algorithms mentioned above range from $\{10^{-4}, ..., 10^4\}$, with the learning rate $\eta = 0.001$, $\sigma = 2$, and $\theta = 0.5$. In addition, for SVMrbf, we set the bandwidth parameter to 1. For RF, the number of decision trees is 100. For KFFUSVM, the regularization parameter is set to 0.5 , the Universum penalty coefficient is 0.1 , the Universum margin threshold is 0.2 , the Universum sample ratio is 0.2 , and the maximum number of iterations is 100. For ASVM, the number of base classifiers is 50, and the maximum number of SVM iterations is 100. For $WCNN_1$, the parameters are configured as follows: The number of networks is set to 100, the hidden layer size is 10, the weight constraint is established at 0.5, the learning rate is specified as 0.01, the number of epochs is defined as 50, and the batch size is determined to be 32. For XGBoost, the base classifier is SVM and the number is 10. For CTSVM and the A-R-U-SQSSVM proposed by us, the truncation

parameter values range from $\{10^{-3}, ..., 10^{0}\}$. For A-R-U-SQSSVM, $\varepsilon > 0$ is a threshold parameter, ε is chosen from {0.001, 0.01, 0.05, 0.1, 0.2}, and parameter $0 < p \leq 2$, p is chosen from {0.1, 0.3, ..., 1.7, 1.9}. In addition, we used 5-fold cross-validation grid search to find the optimal parameters for A-R-U-SQSSVM. In addition to the fixed parameters mentioned above, such as in formula (3.20), there are two regularization parameters C and C_u. We conducted extensive experiments on the Sonar and Hepat datasets from the UCI dataset, and the results are shown in Figure 3.



(a) Sonar. (b) Sonar.

(c) Hepat. (d) Hepat.

Figure 3. Effects of C and C_u on ACC and F_1.

From Figure 3, the following conclusions can be drawn:

For ACC, on the Sonar dataset, the influence of C and C_u on ACC is quite significant. When $C = 10^{-2}$ and $C_u = 10^{-4}$, the ACC is = 72.00%. When $C = 10^{0}$ and $C_u = 10^{-1}$, the ACC is 86.31%. The optimal range of values corresponding to the best ACC is: $C \in [10^{-1}, 10^{2}]$, $C_u \in [10^{-1}, 10^{2}]$. On the Hepat dataset, the impact of C and C_u on ACC is very small. When $C = 10^{3}$ and $C_u = 10^{-1}$, and $C = 10^{4}$ and $C_u = 10^{-1}$, the ACC is 80.00%. When $C = 10^{2}$ and $C_u = 10^{-2}$, and $C = 10^{1}$ and $C_u = 10^{-2}$, the ACC is 86.15%. The optimal range of values corresponding to the best ACC is $C \in [10^{-4}, 10^{2}]$, $C_u \in [10^{-2}, 10^{2}]$.

For F_1, on the Sonar dataset, C and C_u have a significant impact on F_1. When $C = 10^{-4}$ and $C_u = 10^{-3}$, $F_1 = 54.00\%$. When $C = 10^{2}$ and $C_u = 10^{1}$, $F_1 = 78.67\%$. The optimal range of values corresponding to the best F_1 is: $C \in [10^{-1}, 10^{2}]$, $C_u \in [10^{-1}, 10^{2}]$. On the Hepat dataset, C and C_u have a minor impact on F_1. When $C \in [10^{-4}, 10^{0}]$ and $C_u \in [10^{-4}, 10^{4}]$, $F_1 = 71.51\%$. When $C = 10^{1}$ and

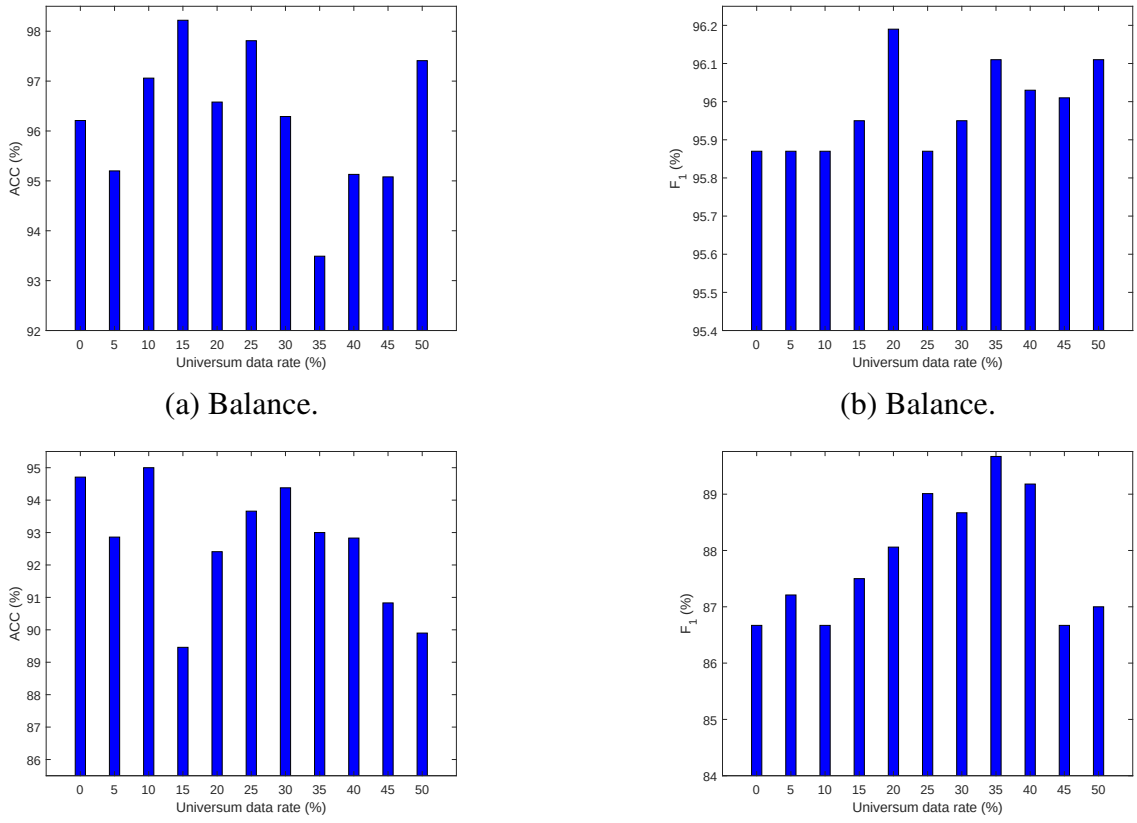
$C_u = 10^0$, $F_1 = 78.67\%$. The optimal range of values corresponding to the best ACC is $C \in [10^1, 10^4]$, $C_u \in [10^{-1}, 10^1]$.

Therefore, in all subsequent experiments, we fix $C = C_u = 1$.

4.2. Experimental results

4.2.1. The influence of Universum data

In order to further study the effect of Universum data on ACC and F_1, we conducted additional experiments using the Balance dataset and Ionosp dataset from the UCI database. Our Universum data generation rules are as follows: First, we control the proportion of randomly generated Universum data. Next, we calculate the amount of Universum data based on the proportion, ensuring an equal number of Universum data in the positive and negative classes. Finally, we construct a new dataset containing Universum data. By controlling the proportion of Universum data, we recorded the ACC and F_1 of our method on the Balance dataset and Ionosp dataset, respectively. The experimental results are shown in Figure 4.



(a) Balance. (b) Balance.

(c) Ionosp. (d) Ionosp.

Figure 4. Effects of Universum data on ACC and F_1.

From Figure 4, we can draw the following conclusions:

The Universum data has a significant impact on the classification performance. As the amount of Universum data increases, ACC and F_1 do not always increase. This indicates that having more

Universum data is not necessarily better. For the Balance dataset, when the ratio of Universum data is 15%, its ACC reaches the highest at 98.22%, and when the ratio is 20%, its F_1 reaches the highest at 96.19%. For the Ionosp dataset, when the ratio of Universum data is 10%, its ACC reaches the highest at 95.00%, and when the ratio is 35%, its F_1 reaches the highest at 89.67%. Therefore, in the subsequent experiments, we fixed the ratio of Universum data at 20%, indicating that the number of Universum data accounts for 20% of the total sample size of the corresponding dataset.

4.2.2. Results on UCI datasets

To study the effectiveness, classification performance, and robustness of the method in this paper compared to eight other algorithms, experiments were conducted on 12 UCI datasets in environments with Gaussian noise levels of 0%, 15%, and 30%. The experimental results are recorded in Tables 4–6, respectively. The visualization of experimental results of nine algorithms on UCI datasets under different Gaussian noise environments is shown in Figure 5. In addition, the highest results are highlighted in bold.

From Table 4, the following conclusions can be drawn: The proposed algorithm A-R-U-SQSSVM (Ours) achieved the highest accuracy (ACC) in 7 out of 12 datasets (Balance, German, QSAR, a8a, a7a, a5a, a3a), with an average ACC of 79.15%, significantly outperforming other models (the second place being KFFUSVM with 76.84%). The F_1 score is also leading, with an average of 86.12%. Additionally, the average running time is 2.05 seconds, slightly faster than XGBoost (3.05 seconds) and RF (3.62 seconds), and much faster than CTSVM (1602.45 seconds) and KFFUSVM (1150.32 seconds). Furthermore, algorithms such as CTSVM, KFFUSVM, XGBoost, and EPCC also performed well, with average ACCs above 75%. The average F_1 scores for RF, KFFUSVM, XGBoost, EPCC, and WCNN_1 are also high, all above 83%; while the average ACCs for SQSSVM and ASVM are relatively low, at 72.15% and 73.12%, respectively. This is sufficient to demonstrate the effectiveness, efficiency, and excellent classification capability of the proposed algorithm.

From Tables 5 and 6, the following conclusions can be drawn: The classification performance of all models decreases with the increase of noise, but the drop for A-R-U-SQSSVM is the smallest. When Gaussian noise increases to 15%, our method performs the best across 8 datasets, with an average accuracy (ACC) of 76.45%, and a drop of only 2.7%, which is less than that of other algorithms (for example, CTSVM has a drop of 2.0%, but with lower baseline performance). Its F_1 is 84.67%, higher than that of the other eight algorithms, and it has the shortest running time. A similar conclusion holds when Gaussian noise increases to 30%. Additionally, on large-scale datasets (a8a, a7a, a5a, a3a, a1a), the classification performance of all algorithms is poor, and the running time increases to varying degrees, with the running times of CTSVM, SQSSVM, and KFFUSVM increasing by hundreds to thousands of times, while the algorithm proposed in this paper has the shortest running time.

Overall, the A-R-U-SQSSVM based on upper-bounded norm distance, generalized Welsch loss function, and AdaBoost is effective, with excellent classification performance and robustness, as well as high efficiency.

Table 4. Experimental results on UCI datasets without Gaussian noise.

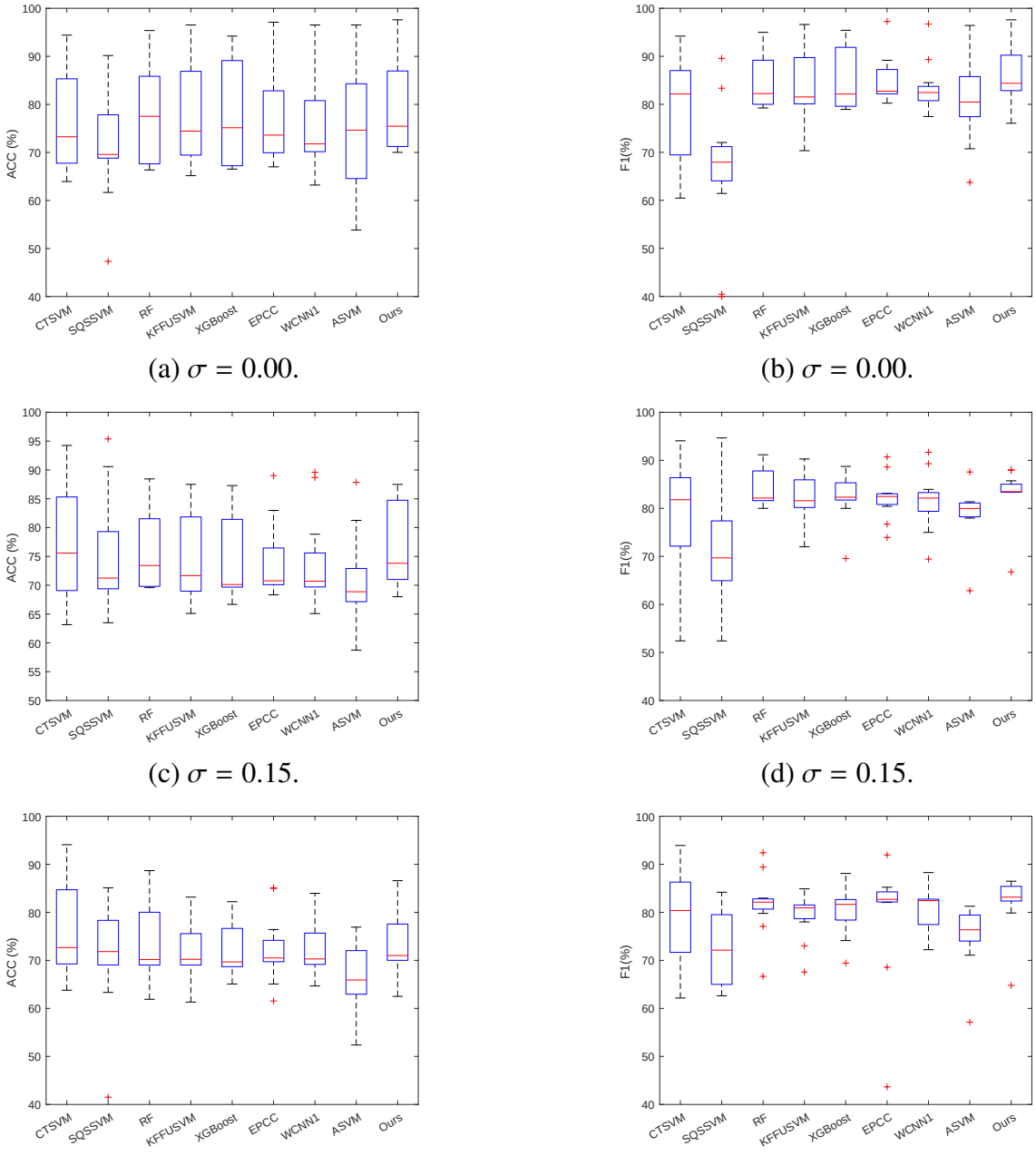
Datasets	CTSVM	SQSSVM	RF	KFFUSVM	XGBoost	EPCC	$WCNN_1$	ASVM	Ours
	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC
	F_1	F_1	F_1	F_1	F_1	F_1	F_1	F_1	F_1
	Times	Times	Times	Times	Times	Times	Times	Times	Times
Balance	94.42	90.17	95.38	96.53	94.22	97.11	96.53	96.53	**97.61**
	94.21	89.57	95.00	96.63	93.98	97.27	96.74	96.43	**97.60**
	4.80	1.12	0.70	1.13	5.48	1.30	3.24	0.91	0.78
Australian	85.22	69.71	85.58	**87.02**	86.06	85.10	82.21	84.62	86.49
	84.50	40.42	84.69	**86.83**	84.97	84.10	79.33	80.95	85.39
	9.36	1.25	1.32	5.71	1.79	0.29	2.89	0.77	0.77
Hepat	78.00	82.98	80.85	82.98	**91.49**	76.60	76.60	76.60	82.00
	85.17	66.67	88.61	89.74	**94.74**	86.75	84.51	84.06	89.23
	1.23	0.08	0.75	0.13	1.71	0.31	0.75	0.97	0.55
German	76.20	61.67	75.67	72.67	75.67	67.00	72.67	72.67	**77.40**
	83.24	61.43	84.03	80.84	83.67	80.24	80.93	81.28	**85.71**
	38.73	2.53	1.46	9.26	2.03	0.50	3.15	1.00	0.69
Ionosp	85.43	68.87	92.45	89.62	**93.40**	83.96	63.21	83.96	90.86
	89.81	69.28	94.44	92.81	**95.42**	89.17	77.46	88.44	93.46
	1.80	0.27	0.57	0.44	1.84	0.31	1.88	0.56	0.94
QSAR	85.78	47.32	86.12	86.75	86.75	81.70	84.86	84.86	**87.43**
	88.89	39.95	89.77	89.71	89.76	87.76	89.33	87.50	**91.24**
	18.54	3.62	0.86	6.83	1.89	0.46	4.69	1.12	1.38
Sonar	67.67	**82.54**	79.37	76.19	74.60	79.37	79.37	77.78	73.50
	68.58	**83.36**	80.00	76.19	78.95	82.19	80.60	80.00	76.05
	1.83	0.12	0.59	0.17	1.03	0.36	1.05	1.07	1.71
a8a	68.71	69.46	66.35	69.26	66.89	70.61	70.88	65.34	**72.00**
	81.06	66.67	79.23	81.84	79.67	82.77	82.96	78.28	**83.35**
	2370.04	1400.33	6.11	1964.23	2.60	36.35	43.22	11.65	2.00
a7a	70.32	70.32	67.50	65.16	66.67	68.70	70.50	63.80	**71.47**
	66.67	66.67	80.00	70.35	79.50	81.45	82.70	76.56	**83.35**
	4964.80	1574.80	9.71	2532.35	5.12	44.24	100.72	31.97	2.34
a5a	66.74	68.74	66.74	**71.00**	66.53	70.12	69.83	53.84	70.01
	79.50	69.22	79.52	81.23	79.40	**82.44**	82.23	63.75	82.35
	5614.67	2014.67	14.67	2512.63	8.61	82.26	86.77	23.22	4.63
a3a	67.83	69.18	67.74	68.66	67.55	70.53	70.44	58.48	**71.00**
	70.34	72.04	80.34	79.35	80.13	**82.72**	82.66	70.74	82.35
	4616.90	2616.90	16.90	3413.10	10.31	148.16	107.66	29.18	4.10
a1a	63.94	**73.12**	67.94	69.62	68.13	69.75	69.87	67.17	70.01
	60.46	70.33	80.46	81.00	80.60	82.18	82.26	79.82	**83.36**
	5717.60	2717.75	17.60	3421.67	11.77	124.49	93.40	35.39	4.67
Avg.(ACC)	75.86	72.15	76.32	76.84	76.12	76.83	76.32	73.12	79.15
Avg.(F_1)	78.44	65.28	83.67	84.12	84.01	84.56	83.98	79.34	86.12
Avg.(Time)	1602.45	856.72	3.62	1150.32	3.05	24.32	33.45	8.76	2.05

Table 5. Experimental results on UCI datasets with 15% Gaussian noise.

Datasets	CTSVM	SQSSVM	RF	KFFUSVM	XGBoost	EPCC	$WCNN_1$	ASVM	Ours
	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC
	F_1	F_1	F_1	F_1	F_1	F_1	F_1	F_1	F_1
	Times	Times	Times	Times	Times	Times	Times	Times	Times
Balance	94.25	**95.38**	88.44	84.97	87.28	89.02	89.60	87.86	87.49
	94.06	**94.67**	88.64	84.88	86.90	88.62	89.29	87.57	87.92
	5.42	0.91	0.99	1.50	2.21	0.28	1.74	0.95	0.29
Australian	85.22	75.00	82.21	**87.50**	84.13	79.33	78.85	81.25	85.35
	84.50	63.23	80.63	**87.00**	81.56	73.94	75.00	80.00	84.32
	9.06	1.21	1.64	3.85	2.85	0.39	3.52	0.80	0.42
Hepat	74.83	72.34	80.85	78.72	78.72	**82.98**	72.34	70.21	77.33
	81.75	66.67	89.41	88.10	88.10	**90.70**	83.95	81.08	85.74
	1.97	0.07	0.68	0.13	2.01	0.28	0.78	1.03	0.22
German	76.30	69.33	70.67	72.33	69.00	68.33	71.00	69.33	**76.60**
	83.49	69.14	81.20	81.35	80.00	81.19	79.63	78.20	**84.27**
	36.83	2.39	1.98	5.50	2.96	0.52	5.93	1.07	0.45
Ionosp	86.00	**90.57**	87.74	86.79	84.91	73.58	88.68	74.53	84.14
	90.16	90.12	91.16	90.28	88.73	83.13	**91.67**	81.38	88.01
	1.93	0.25	0.66	0.33	1.79	0.39	1.50	0.71	0.73
QSAR	85.40	83.60	80.76	77.29	76.97	70.35	72.24	71.29	**86.62**
	88.27	81.89	86.94	83.71	83.74	80.42	79.15	77.97	83.60
	22.15	2.83	1.11	6.07	1.55	0.45	4.79	1.61	1.13
Sonar	**79.83**	63.49	76.19	66.67	66.67	73.02	65.08	58.73	68.00
	79.76	30.53	80.00	72.00	69.57	76.71	69.44	62.86	66.77
	1.44	0.11	0.54	0.15	1.08	0.34	1.02	1.14	1.88
a8a	68.71	69.63	69.63	69.26	69.46	69.16	69.76	67.23	**71.01**
	81.84	66.67	82.04	81.84	81.90	81.77	82.18	79.95	**83.35**
	2464.23	1427.48	7.48	1899.65	5.05	26.07	51.84	11.48	2.40
a7a	69.44	69.44	69.71	65.10	70.12	70.88	69.63	67.06	**71.00**
	52.41	52.41	82.08	72.14	82.36	82.96	82.10	79.62	**83.34**
	5114.33	1511.33	14.33	2477.09	10.46	52.99	82.87	22.04	2.52
a5a	68.12	70.14	69.89	**71.00**	69.94	70.66	70.39	65.48	**71.00**
	72.23	70.23	82.23	81.23	82.27	82.81	82.62	78.27	**83.35**
	5623.61	1623.61	23.61	2301.06	17.79	77.86	78.47	24.40	5.06
a3a	69.58	69.18	69.58	68.66	70.12	69.84	69.50	68.41	**71.01**
	72.04	72.04	82.04	79.35	82.40	82.24	82.00	81.16	**83.36**
	4827.90	2827.90	27.90	3581.75	23.26	122.92	94.57	35.09	5.75
a1a	63.15	**73.15**	70.15	70.00	69.88	70.52	70.13	67.42	71.00
	62.44	72.90	82.44	81.00	82.23	82.71	82.44	80.07	**83.35**
	5230.33	3200.69	30.31	2844.39	25.32	128.55	118.23	32.94	4.39
Avg.(ACC)	73.89	72.54	73.12	74.23	73.45	74.02	73.68	70.89	76.45
Avg.(F_1)	76.12	63.45	81.34	82.67	82.01	83.12	82.45	77.89	84.67
Avg.(Time)	1602.45	856.72	3.62	1150.32	3.05	24.32	33.45	8.76	2.05

Table 6. Experimental results on UCI datasets with 30% Gaussian noise.

Datasets	CTSVM	SQSSVM	RF	KFFUSVM	XGBoost	EPCC	$WCNN_1$	ASVM	Ours
	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC
	F_1	F_1	F_1	F_1	F_1	F_1	F_1	F_1	F_1
	Times	Times	Times	Times	Times	Times	Times	Times	Times
Balance	**94.08**	75.49	79.19	78.61	74.57	84.97	76.88	73.41	85.42
	93.92	80.62	79.78	80.42	74.12	85.23	77.53	74.16	85.63
	4.85	0.83	0.97	1.22	1.66	0.30	1.53	0.94	0.37
Australian	85.22	85.10	84.13	83.17	82.21	61.54	79.81	76.92	**86.59**
	84.50	84.19	82.16	81.08	78.86	43.66	77.66	71.08	**85.49**
	9.93	1.04	1.26	3.48	2.07	0.45	2.76	0.91	0.60
Hepat	77.46	**85.11**	80.85	70.21	78.72	**85.11**	74.47	61.70	75.33
	83.43	66.67	89.41	82.50	88.10	**91.95**	84.62	75.00	85.33
	3.08	0.06	0.75	0.16	2.23	0.29	0.64	0.68	0.29
German	75.00	63.33	64.33	75.67	73.00	71.67	67.67	65.67	**75.70**
	82.51	63.33	77.09	84.89	83.16	83.50	77.39	75.76	**86.48**
	36.06	3.00	1.13	5.93	2.94	0.42	5.51	0.89	0.59
Ionosp	84.57	41.51	**88.68**	75.47	81.13	76.42	83.96	73.58	79.43
	89.16	20.51	**92.41**	80.88	86.11	84.85	88.28	80.00	84.92
	2.48	0.28	0.53	0.39	1.85	0.34	1.37	0.68	0.55
QSAR	**84.83**	75.71	73.82	69.40	66.56	71.92	64.67	70.66	70.19
	88.09	78.41	82.96	78.00	77.92	83.67	74.77	79.91	79.87
	25.67	2.59	1.36	6.59	1.47	0.52	3.83	1.91	1.32
Sonar	69.60	**80.95**	61.90	61.90	65.08	65.08	68.25	52.38	62.50
	71.27	**81.05**	66.67	67.57	69.44	68.57	72.22	57.14	64.78
	1.45	0.12	0.49	0.17	1.18	0.36	0.95	0.96	1.16
a8a	68.71	70.54	70.54	70.22	68.68	70.47	**70.84**	61.45	70.01
	78.24	62.62	82.62	81.84	81.31	82.68	82.93	73.87	**83.35**
	2568.18	1568.18	8.18	1911.00	5.25	24.22	55.96	9.95	1.86
a7a	**70.32**	**70.32**	69.08	61.32	68.70	70.34	70.30	64.43	70.01
	66.67	66.67	81.63	73.02	81.40	82.59	82.55	77.26	**82.96**
	4881.70	1488.70	13.70	2452.60	9.95	45.56	57.77	20.94	2.77
a5a	68.91	68.91	68.97	**71.00**	69.80	69.57	70.21	64.21	**71.00**
	72.13	72.66	81.57	81.23	82.16	82.06	82.49	76.96	**82.35**
	5625.34	1625.34	25.34	2394.33	18.51	91.60	80.52	24.70	4.33
a3a	69.65	69.18	69.65	68.66	69.42	70.53	70.06	68.67	**71.01**
	72.08	72.08	82.08	79.35	81.92	**82.72**	82.39	81.27	82.35
	5528.99	2528.99	28.99	2477.26	21.32	103.87	113.47	32.50	4.26
a1a	63.79	**73.12**	69.79	69.62	69.54	69.87	70.29	66.17	70.01
	62.16	72.16	82.16	81.00	81.99	82.26	82.55	78.96	**82.36**
	4631.98	2611.98	31.36	3019.47	27.27	111.08	133.03	35.27	3.47
Avg.(ACC)	72.45	71.23	71.89	73.12	72.34	73.45	72.89	69.12	75.12
Avg.(F_1)	74.89	61.23	79.45	80.12	80.34	81.23	80.89	75.45	82.89
Avg.(Time)	1702.34	923.45	4.45	1256.78	4.56	28.12	39.45	10.12	2.67



(a) σ = 0.00. (b) σ = 0.00.

(c) σ = 0.15. (d) σ = 0.15.

(e) σ = 0.30. (f) σ = 0.30.

Figure 5. Experimental results of nine algorithms on UCI datasets under different Gaussian noise environments.

4.3. Statistical test analysis

In this section, to further compare the performance differences between this algorithm and the other eight algorithms in a fair and objective manner, this section uses the Friedman test [56]. The underlying assumption of this statistical test is that all algorithms perform equally.

In the Friedman test, this section lists Avg.rank (ACC) and Avg.rank (F_1) with 9 algorithms computed on 12 datasets in Table 7. In addition, the parameters of the statistical variables for

Friedman's test are calculated by the following formula:

$$X_F^2 = \frac{12N}{k(k+1)}[\sum_j R_j^2 - \frac{k(k+1)^2}{4}], \quad F_F((k-1),(k-1)(N-1)) = \frac{(N-1)X_F^2}{N(k-1)-X_F^2},$$

where $N = 12$ represents the number of UCI datasets, $k = 9$ represents the number of algorithms, and R_j represents the average ranking of j methods. In addition, $F_F((k-1),(k-1)(N-1))$ obey f distribution , its degrees of freedom, respectively, $(k-1)$, and $(k-1)(N-1)$. Get X_F^2 and F_F^2, when get $\alpha = 0.05$, get $F_{\alpha=0.1}(8, 88) = 2.0454$.

From Table 8, it can be concluded that $F_F > F_\alpha$ applies to all cases. Therefore, the rejection of null hypothesis indicates that there is a significant performance difference between the algorithm in this paper and the other eight algorithms.

Table 7. The average ACC ranking and average F_1ranking of the nine algorithms on the UCI dataset.

Gaussian noise		CTSVM	SQSSVM	RF	KFFUSVM	XGBoost	EPCC	$WCNN_1$	ASVM	Ours
$\sigma = 0.00$	ACCÅÅ	5.83	5.42	5.17	4.17	5.17	4.79	4.86	7.25	2.33
	F_1ÅÅ	6.63	7.96	4.21	4.50	4.42	3.92	4.25	6.88	2.25
$\sigma = 0.15$	ACCÅÅ	4.75	4.92	4.08	5.13	5.33	5.00	5.04	7.96	2.79
	F_1ÅÅ	4.96	7.58	3.38	4.92	4.46	4.42	4.63	7.42	3.25
$\sigma = 0.30$	ACCÅÅ	3.88	4.42	4.54	6.00	5.88	4.08	4.50	8.08	3.63
	F_1ÅÅ	5.17	6.92	4.25	5.83	5.17	3.50	4.00	7.25	2.92

Table 8. The results of Friedman test.

Gaussian noise	ACC		F_1		
	X_F^2	F_F	X_F^2	F_F	F_α
$\sigma = 0.00$	22.0274	3.2755	41.0477	8.2167	2.0454
$\sigma = 0.15$	23.5014	3.5658	30.6762	5.1656	2.0454
$\sigma = 0.30$	25.8184	4.0467	28.3752	4.6156	2.0454

5. Conclusions

In this paper, we introduce the capped $L_{2,p}$-norm distance metric and a generalized adaptive loss function into the SQSSVM, replacing the L_2-norm and misclassification terms. This yields a novel, robust, and adaptive R-SQSSVM. We then utilize the prior knowledge of Universum data points to obtain the R-U-SQSSVM model, which exhibits good generalization performance. Finally, by incorporating the AdaBoost method into our R-U-SQSSVM model, we obtain the final A-R-U-SQSSVM model. We transform our model into a quadratic programming problem using the concave duality theorem and the HQ optimization algorithm. This transformation allows for effective solutions through convex optimization methods such as SMO. Experimental results on both synthetic and UCI datasets demonstrate that our proposed A-R-U-SQSSVM exhibits excellent classification performance and robustness, outperforming eight other advanced algorithms.

In future work, we plan to expand our method to incorporate semi-supervised learning and deep learning techniques to further investigate the performance of the proposed A-R-U-SQSSVM and

enhance our theoretical framework. Furthermore, we will explore alternative loss functions and hybrid enhancement techniques to optimize algorithm performance. We also intend to combine A-R-U-SQSSVM with deep learning architectures for feature extraction and classification of complex datasets. Lastly, we aim to improve explainability in high-risk domains through the implementation of explainable artificial intelligence (AI) frameworks.

Author contributions

Bao Ma: Conceptualization, methodology, software, writing–original draft; Yanrong Ma: Software, validation, supervision, project administration; Jun Ma: Software, validation, writing–review and editing, supervision, project administration. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

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

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (No.62366001, No.12361062), in part by the Natural Science Foundation of Ningxia Provincial (No.2024AAC05055, No.2023AAC02053, No.2022AAC03286), in part by the Fundamental Research Funds for the Central Universities of North Minzu University (No.2023ZRLG01, No.2021JCYJ07), in part by the Graduate Innovation Project of North Minzu University (YCX24089).

Conflict of interest

There are no conflicts of interest in this study.

References

1. Y. Bhattarai, S. Duwal, S. Sharma, R. Talchabhadel, Leveraging machine learning and open-source spatial datasets to enhance flood susceptibility mapping in transboundary river basin, *Int. J. Digit. Earth*, **17** (2024), 2313857. https://doi.org/10.1080/17538947.2024.2313857
2. W. Liu, D. Tao, Multiview hessian regularization for image annotation, *IEEE T. Image Process.*, **22** (2013), 2676–2687. https://doi.org/10.1109/TIP.2013.2255302
3. Q. Yuan, H. Shen, T. Li, Z. Li, S. Li, Y. Jiang, et al., Deep learning in environmental remote sensing: Achievements and challenges, *Remote Sens. Environ.*, **241** (2020), 111716. https://doi.org/10.1016/j.rse.2020.111716
4. C. Cortes, V. Vapnik, Support-vector networks, *Mach. Learn.*, **20** (1995), 273–297. https://doi.org/10.1007/BF00994018
5. J. Platt, *Sequential minimal optimization: A fast algorithm for training support vector machines*, Published by Microsoft, 1998.

6. R. E. Fan, P. H. Chen, C. J. Lin, T. Joachims, Working set selection using second order information for training support vector machines, *J. Mach. Learn. Res.*, **6** (2005).
7. Y. Wu, Y. Liu, Robust truncated hinge loss support vector machines, *J. Am. Stat. Assoc.*, **102** (2007), 974–983. https://doi.org/10.1198/016214507000000617
8. L. Wang, H. Jia, J. Li, Training robust support vector machine with smooth ramp loss in the primal space, *Neurocomputing*, **71** (2008), 3020–3025. https://doi.org/10.1016/j.neucom.2007.12.032
9. X. Huang, L. Shi, J. A. Suykens, Support vector machine classifier with pinball loss, *IEEE T. Pattern Anal.*, **36** (2013), 984–997. https://doi.org/10.1109/TPAMI.2013.178
10. X. Shen, L. Niu, Z. Qi, Y. Tian, Support vector machine classifier with truncated pinball loss, *Pattern Recogn.*, **68** (2017), 199–210. https://doi.org/10.1016/j.patcog.2017.03.011
11. C. Yuan, L. Yang, Capped $L_{2,P}$-norm metric based robust least squares twin support vector machine for pattern classification, *Neural Networks*, **142** (2021), 457–478. https://doi.org/10.1016/j.neunet.2021.06.028
12. J. Zhang, Z. Lai, H. Kong, L. Shen, Robust twin bounded support vector classifier with manifold regularization, *IEEE T. Cybernetics*, **53** (2022), 5135–5150. https://doi.org/10.1109/TCYB.2022.3160013
13. J. Ke, C. Gong, T. Liu, L. Zhao, J. Yang, D. Tao, Laplacian Welsch regularization for robust semisupervised learning, *IEEE T. Cybernetics*, **52** (2020), 164–177. https://doi.org/10.1109/TCYB.2019.2953337
14. J. Ma, L. Yang, Q. Sun, Capped L_1-norm distance metric-based fast robust twin bounded support vector machine, *Neurocomputing*, **412** (2020), 295–311. https://doi.org/10.1016/j.neucom.2020.06.053
15. Z. Y. Wang, H. C. So, A. M. Zoubir, Low-rank tensor completion via novel sparsity-inducing regularizers, *IEEE T. Signal Process.*, **72** (2024), 3519–3534. https://doi.org/10.1109/TSP.2024.3424272
16. Z. Y. Wang, H. C. So, *Robust matrix completion via novel M-estimator functions*, In: 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET), Australia: IEEE, 2024. https://doi.org/10.1109/ICECET61485.2024.10698047
17. Z. Y. Wang, H. C. So, A. M. Zoubir, Robust low-rank matrix recovery via hybrid ordinary-Welsch function, *IEEE T. Signal Process.*, **71** (2023), 2548–2563. https://doi.org/10.1109/TSP.2023.3290353
18. Z. Y. Wang, X. P. Li, H. C. So, Robust matrix completion based on factorization and truncated-quadratic loss function, *IEEE T. Circ. Syst. Vid.*, **33** (2022), 1521–1534. https://doi.org/10.1109/TCSVT.2022.3214583
19. A. Eslam, M. G. Abdelfattah, E. S. M. El-Kenawy, H. El-Din Moustafa, Classification of monkeypox using Greylag Goose Optimization (GGO) algorithm, *Fusion Pract. Appl.*, **16** (2024).
20. E. S. M. El-Kenawy, F. H. Rizk, A. M. Zaki, M. E. Mohamed, A. Ibrahim, A. A. Abdelhamid, et al., Football optimization algorithm (fboa): A novel metaheuristic inspired by team strategy dynamics, *J. Artif. Intell. Metaheuristics*, **8** (2024), 21–38. https://doi.org/10.54216/JAIM.080103

21. V. Vijayalakshmi, M. S. Babu, R. P. Lakshmi, *Kfcm algorithm for effective brain stroke detection through SVM classifier*, In: 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA), India: IEEE, 2018. https://doi.org/10.1109/ICSCAN.2018.8541179

22. A. Kumar, G. Sharma, R. Pareek, S. Sharma, P. Dadheech, M. K. Gupta, Performance optimisation of face recognition based on LBP with SVM and random forest classifier, *Int. J. Biometrics*, **15** (2023), 389–408. https://doi.org/10.1504/IJBM.2023.130644

23. V. Vapnik, *Estimation of dependences based on empirical data*, Springer Science and Business Media, 2006.

24. J. Weston, R. Collobert, F. Sinz, L. Bottou, V. Vapnik, *Inference with the universum*, In: Proceedings of the 23rd international conference on Machine learning, 2006, 1009–1016. https://doi.org/10.1145/1143844.1143971

25. C. Shen, P. Wang, F. Shen, H. Wang, U Boost: Boosting with the Universum, *IEEE T. Pattern Anal.*, **34** (2011), 825–832. https://doi.org/10.1109/TPAMI.2011.240

26. X. Yan, H. Zhu, A kernel-free fuzzy support vector machine with Universum, *J. Ind. Manag. Optim.*, **19** (2023). https://doi.org/10.3934/jimo.2021184

27. H. Moosaei, A. Mousavi, M. Hladík, Z. Gao, Sparse L_1-norm quadratic surface support vector machine with Universum data, *Soft Comput.*, **27** (2023), 5567–5586. https://doi.org/10.1007/s00500-023-07860-3

28. X. Bai, V. Cherkassky, *Gender classification of human faces using inference through contradictions*, In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong: IEEE, 2008, 746–750. https://doi.org/10.1109/IJCNN.2008.4633879

29. C. L. Liu, W. H. Hsaio, C. H. Lee, T. H. Chang, T. H. Kuo, Semi-supervised text classification with universum learning, *IEEE T. Cybernetics*, **46** (2015), 462–473. https://doi.org/10.1109/TCYB.2015.2403573

30. B. Richhariya, M. Tanveer, A. H. Rashid, Alzheimer's disease neuroimaging initiative, diagnosis of Alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE), *Biomed. Signal Proces.*, **59** (2020), 101903. https://doi.org/10.1016/j.bspc.2020.101903

31. I. Dagher, Quadratic kernel-free non-linear support vector machine, *J. Global Optim.*, **41** (2008), 15–30. https://doi.org/10.1007/s10898-007-9162-0

32. J. Luo, S. C. Fang, Z. Deng, X. Guo, Soft quadratic surface support vector machine for binary classification, *Asia Pac. J. Oper. Res.*, **33** (2016), 1650046. https://doi.org/10.1142/S0217595916500469

33. Y. Bai, X. Han, T. Chen, H. Yu, Quadratic kernel-free least squares support vector machine for target diseases classification, *J. Comb. Optim.*, **30** (2015), 850–870. https://doi.org/10.1007/s10878-015-9848-z

34. X. Yan, H. Zhu, A novel robust support vector machine classifier with feature mapping, *Knowl.-Based Syst.*, **257** (2022), 109928. https://doi.org/10.1016/j.knosys.2022.109928

35. Z. Gao, S. C. Fang, J. Luo, N. Medhin, A kernel-free double well potential support vector machine with applications, *Eur. J. Oper. Res.*, **290** (2021), 248–262. https://doi.org/10.1016/j.ejor.2020.10.040

36. Z. Y. Chen, Z. P. Fan, M. Sun, A multi-kernel support tensor machine for classification with multitype multiway data and an application to cross-selling recommendations, *Eur. J. Oper. Res.*, **255** (2016), 110–120. https://doi.org/10.1016/j.ejor.2016.05.020

37. J. Zhou, Y. Tian, J. Luo, Q. Zhai, A kernel-free Laplacian quadratic surface optimal margin distribution machine with application to credit risk assessment, *Appl. Soft Comput.*, **133** (2023), 109931. https://doi.org/10.1016/j.asoc.2022.109931

38. Z. Gao, Y. Wang, M. Huang, J. Luo, S. Tang, A kernel-free fuzzy reduced quadratic surface v-support vector machine with applications, *Appl. Soft Comput.*, **127** (2022), 109390. https://doi.org/10.1016/j.asoc.2022.109390

39. J. R. Magnus, H. Neudecker, The elimination matrix: Some lemmas and applications, *SIAM J. Algebr. Discrete Meth.*, **1** (1980), 422–449. https://doi.org/10.1137/0601049

40. Z. Qi, Y. Tian, Y. Shi, Twin support vector machine with universum data, *Neural Networks*, **36** (2012), 112–119. https://doi.org/10.1016/j.neunet.2012.09.004

41. S. Dhar, V. Cherkassky, *Cost-sensitive Universum-svm*, In: 2012 11th International Conference on Machine Learning and Applications, USA: IEEE, 2012. https://doi.org/10.1109/ICMLA.2012.45

42. V. Cherkassky, F. M. Mulier, *Learning from data: Concepts, theory, and methods*, John Wiley and Sons, 2007. https://doi.org/10.1002/9780470140529

43. R. He, B. Hu, X. Yuan, L. Wang, *M-estimators and half-quadratic minimization*, Robust Recognition via Information Theoretic Learning, Springer, Cham, 2014, 3–11. https://doi.org/10.1007/978-3-319-07416-0_2

44. B. Ma, J. Ma, G. Yu, A novel robust metric distance optimization-driven manifold learning framework for semi-supervised pattern classification, *Axioms*, **12** (2023), 737. https://doi.org/10.3390/axioms12080737

45. T. Zhang, Analysis of multi-stage convex relaxation for sparse regularization, *J. Mach. Learn. Res.*, **11** (2010).

46. R. E. Schapire, The strength of weak learnability, *Mach. Learn.*, **5** (1990), 197–227. https://doi.org/10.1007/BF00116037

47. Y. Freund, R. E. Schapire, *A desicion-theoretic generalization of on-line learning and an application to boosting*, In: European conference on computational learning theory, Berlin, Heidelberg: Springer, 1995, 23–37. https://doi.org/10.1007/3-540-59119-2_166

48. J. Wang, P. Li, R. Ran, Y. Che, Y. Zhou, A short-term photovoltaic power prediction model based on the gradient boost decision tree, *Appl. Sci.*, **8** (2018), 689. https://doi.org/10.3390/app8050689

49. Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, J. Peng, *XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud*, In: 2018 IEEE international conference on big data and smart computing (bigcomp), IEEE, 2018, 251–256.

50. J. Xu, Q. Wu, J. Zhang, Z. Tang, Exploiting Universum data in AdaBoost using gradient descent, *Image Vision Comput.*, **32** (2014), 550–557. https://doi.org/10.1016/j.imavis.2014.04.009

51. C. L. Liu, W. H. Hsaio, C. H. Lee, T. H. Chang, T. H. Kuo, Semi-supervised text classification with universum learning, *IEEE T. Cybernetics*, **46** (2015), 462–473. https://doi.org/10.1109/TCYB.2015.2403573

52. B. Liu, R. Huang, Y. Xiao, J. Liu, K. Wang, L. Li, et al., Adaptive robust Adaboost-based twin support vector machine with universum data, *Inform. Sciences*, **609** (2022), 1334–1352. https://doi.org/10.1016/j.ins.2022.07.155

53. J. Ma, G. Yu, W. Xiong, X. Zhu, Safe semi-supervised learning for pattern classification, *Eng. Appl. Artif. Intel.*, **121** (2023), 106021. https://doi.org/10.1016/j.engappai.2023.106021

54. L. Li, Q. Hu, X. Wu, D. Yu, Exploration of classification confidence in ensemble learning, *Pattern Recogn.*, **47** (2014), 3120–3131. https://doi.org/10.1016/j.patcog.2014.03.021

55. I. E. Livieris, L. Iliadis, P. Pintelas, On ensemble techniques of weight-constrained neural networks, *Evol. Syst.*, **12** (2021), 155–167. https://doi.org/10.1007/s12530-019-09324-2

56. J. Demišar, Statistical comparisons of classifiers over multiple data sets, *J. Mach. Learn. Res.*, **7** (2006), 1–30.

AIMS Press