Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. *, No. *, *, pp. 1-15
DOI:10.22034/cmde.2025.66530.3113

Modified Newton's method for solving parametric ν-support vector regression with Universum data

Fatemeh Bazikar[1], **Bahram Sadeghi Bigham**[1,*], **and Atefeh Hemmati**[2]

[1]Department of Computer Science, Faculty of Mathematical Sciences, Alzahra University, Tehran, Iran.

[2]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.

Abstract

Universum, representing a third category distinct from the two primary classes in classification tasks, facilitates the incorporation of prior knowledge into the learning process. Extensive studies have confirmed its effectiveness in improving both supervised and semi-supervised classification models. Recently, Universum data has been introduced into parametric ν-support vector regression (UPar-ν-SVR) to enhance generalization performance. In this paper, we present a Newton-based method for solving UPar-ν-SVR, with the objective of further improving its efficiency and accuracy. Our approach reformulates the problem into an unconstrained convex optimization framework and employs a generalized Newton's method for its solution. To assess the effectiveness of our proposed method, we conduct comprehensive experiments on multiple UCI benchmark data sets. The experimental results indicate that our algorithm outperforms existing techniques, providing superior generalization capabilities and computational efficiency.

Keywords. Parametric $\nu-$support vector regression, Univesum data, Newton's method.

2010 Mathematics Subject Classification. 65L05, 34K06, 34K28.

1. Introduction

Supervised learning, a key domain within machine learning, offers robust methods for classifying data sets, enabling solutions to a variety of practical issues encountered in the real world. These applications span diverse areas such as pinpointing signal origins, diagnosing cardiovascular conditions, categorizing images, detecting lung and colon cancers, analyzing textual data, and enhancing agricultural practices, among others [1, 3, 7, 15, 17, 29, 48].

The versatility of supervised learning has garnered considerable interest from the research community. Among its prominent techniques, the support vector machine (SVM) stands out, initially developed by Vapnik and Chervonenkis [46] in 1974. Designed originally for tasks involving classification and regression, SVM has become a cornerstone for classification challenges. It solves these issues by defining a convex quadratic programming problem, aiming to create two parallel hyperplanes that maximize the distance between classes. Subsequent research ushered in classifiers utilizing non-parallel hyperplanes, capturing considerable interest from the academic community. In 2007, Khemchandani and Chandra [18] presented a binary classification method relying on non-parallel hyperplanes named twin support vector machine (TSVM). TSVM's share a core idea: they build two non-parallel hyperplanes, each aligned nearer to its associated class while ensuring the greatest possible distance from the opposing class. Although TSVM's optimization structure resembles SVM's, it tackles two compact quadratic programming problems (QPPs). As a result, TSVM outperforms traditional SVM by delivering a learning pace nearly four times quicker [18]. Thanks to its reduced computational demands, TSVM has emerged as a widely adopted machine learning strategy for classification purposes.

Numerous researchers have investigated diverse modifications of TSVM across supervised, semi-supervised, and unsupervised learning frameworks, as well as their real-world uses [6, 19, 30, 38, 42, 51]. As an illustration, Shao et

Received: 24 March 2025 ; Accepted: 16 July 2025.
∗ Corresponding author. Email: b_sadeghi_b@alzahra.ac.ir.

al. [38] presented the twin bounded support vector machine (TBSVM), a refined version of TSVM, by embedding regularization elements into its objective functions. This adjustment broadened TSVM's applicability and elevated its classification accuracy. In a similar vein, Peng [31] put forward the ν-twin support vector machine (ν-TSVM), incorporating two extra parameters to adjust the equilibrium between support vectors and margin discrepancies. Additionally, Peng [32] proposed the twin parametric-margin support vector machine (TPMSVM), designed to construct two non-parallel hyperplanes that define the parametric-margin boundary for classification purposes.

Advancing these innovations, Tanveer et al. [41] introduced the large-scale pinball twin support vector machine (LPTWSVM), tailoring it for handling expansive data sets and conducting a thorough evaluation of different TSVM approaches. Moosaei et al. [24] reimagined TSVM as a linear programming challenge by weaving L_1 and L_∞ norms into its objective functions, and they proposed the generalized Newton's method (GTWSVM) to adeptly tackle the unconstrained optimization task. While TSVM enjoys widespread acclaim for binary classification and has attracted considerable interest, it struggles to derive kernel-based models directly using primal formulations and duality principles. Nevertheless, TSVM and its derivatives come with certain shortcomings. For example, TSVM demands the calculation of matrix inverses before model training, rendering it less effective when handling vast data sets. Determining these inverses through traditional techniques can prove highly challenging or even unfeasible in such scenarios. Moreover, a further drawback shared by classical SVM, TSVM, and their advanced iterations is their presumption of consistent noise throughout the data set. Yet, in practical contexts like activity recognition, noise levels frequently vary depending on feature characteristics, as highlighted by Tanveer et al. [40]. Additionally, SVM frameworks and their practical uses have been explored to refine TSVM approaches, particularly in addressing regression challenges, which find relevance across multiple domains [2, 52].

Support vector regression (SVR), a derivative of the SVM, efficiently tackles regression challenges. SVM approaches binary classification by casting it as a convex optimization task, as outlined by Vapnik [45]. These tasks seek to identify a dividing hyperplane that optimizes the margin while maintaining precise class separation. In SVM, support vectors define the ideal hyperplanes. The method's ability to deliver sparse solutions and strong generalization makes it exceptionally well-suited for regression applications. To adapt SVM into SVR, an ϵ-insensitive zone, known as the ϵ-tube, is established around the function. This approach reformulates the optimization problem as finding a tube that best aligns with the continuous-valued function, ensuring a trade-off between model complexity and predictive accuracy. A notable theoretical leap in SVR was achieved by Jianlin et al. [16], who introduced a clear geometric perspective for SVR, illustrating how SVR and SVM could be linked by reinterpreting SVR as a classification issue. Other advanced techniques in data classification include the parametric ν-support vector machine (Par-ν-SVM) and parametric ν-support vector regression (Par-ν-SVR) [12]. In contrast to traditional ν-SVM and SVM, which utilize parallel hyperplanes, Par-ν-SVM differentiates classes by maximizing the margin between two non-parallel hyperplanes. Par-ν-SVR, a robust approach for both classification and regression, effectively overcomes the challenges associated with non-parallel methods, such as solving dual optimization problems and computing matrix inverses before training. Furthermore, this method addresses the issue of uniform noise in data sets by integrating an an-isotropic noise structure, enhancing its adaptability to complex data distributions. This method leads to a quadratic programming problem, providing a framework for jointly learning margin determination and function approximation. In recent research, Ketabchi et al. [17] explored the Par-ν-SVR approach for classification tasks and proposed an innovative solution to its primal problem. They restructured the Par-ν-SVR model into an unconstrained convex optimization problem and employed an enhanced version of Newton's method for its solution. The integration of Universum data in supervised and semi-supervised learning has garnered significant interest. Initially introduced by Weston et al. [47], Universum data represents instances that do not belong to any specific class. Their study explored its application within the support vector machine (SVM) framework, leading to the development of the Universum support vector machine (U-SVM). The findings indicated that U-SVM outperformed conventional SVM models that do not utilize Universum data. Building on this concept, several extensions of traditional models have been introduced. Chapelle et al. [4] proposed a least squares version of the Universum support vector machine. Later, Qi et al. [33] integrated Universum data into the twin support vector machine (TSVM), resulting in the Universum twin support vector machine (U-TSVM), which outperformed models that excluded Universum data. Building on this progress, Xu et al. [50], developed a least squares variant of TSVM with Universum data (ULS-TSVM) to further improve the effectiveness

of U-TSVM. In EEG signal classification, several studies have proposed novel methodologies leveraging U-SVM. For instance, Richhariya and Tanveer [34] applied U-SVM for EEG signal classification Gupta et al. [11] developed a regularized Universum twin support vector machine (RUTWSVM) by incorporating regularization constraints into the U-SVM framework, enhancing its stability and robustness. Kumar and Gupta [20] proposed the Universum twin bounded Lagrangian support vector machine (ULTBSVM) to differentiate between healthy and seizure EEG signals. Similarly, Ganaie et al. [9] presented a Pin-Ball Universum twin support vector machine (Pin-UTSVM) for EEG signal classification, further expanding the applicability of Universum-based models in this domain. Richhariya and Tanveer [36] introduced the Universum twin support vector machine with least squares and parametric margin (ULSTPMSVM), which builds upon the least squares parametric margin twin support vector machine (LSTPMSVM) developed by Shao et al. [39]. Additionally, Richhariya et al. [37] proposed an angle-based Universum twin support vector machine (AULSTSVM) for classification. Moosaei and Hladík [25] presented a Lagrangian-based method for the bounded Universum twin support vector machine. Drawing inspiration from previous research and leveraging the benefits of Universum data, Hazarika et al. [13] introduced a novel TPMSVM model, applying it to EEG signal classification. Furthermore, Xiao et al. [49] explored the concept of Universum learning by utilizing non-target task data as prior knowledge. They introduced a novel approach called the multi-task Universum support vector machine (U-MTLSVM) to enhance learning across multiple tasks. Moosaei et al. [26] introduced UPar-ν-SVR, integrating Universum data into parametric ν-support vector regression to enhance classification performance. They proposed two solutions: Dual Problem Formulation, which improves efficiency by solving the dual problem, and Least Squares Version (LS-UPar-ν-SVR), which reduces complexity by using the L_2 norm and replacing inequality constraints with equality constraints, enabling linear equation solving instead of quadratic programming. Their approach offers an efficient solution for large-scale classification tasks while minimizing computational costs. Universum models have attracted considerable interest from researchers because of their straightforward structure and consistently strong performance in a wide range of domains [10, 21, 23, 27, 28, 35, 53].

In this paper, we present a Newton-based method for solving UPar-ν-SVR, with the objective of further improving its efficiency and accuracy. Our approach reformulates the problem into an unconstrained convex optimization framework and employs a generalized Newton's method for its solution.

The rest of the paper is structured as follows: Section 2 briefly introduces the background of Par-ν-SVR and UPar-ν-SVR. Section 3 describes the NUPar-ν-SVR model and outlines the approach for solving the problem using Newton's method. Section 4 presents experimental results for UCI benchmark data sets. Finally, Section 5 offers the conclusion of the paper.

2. Background

In this section, we provide a brief overview of the Par-ν-SVR and UPar-ν-SVR formulations for binary classification problems. Consider the given training data set χ, which is represented as:

$$\chi = \Big\{(x_1, y_1), \ldots, (x_m, y_m)\Big\} \in (R^n \times \{\pm 1\})^m,$$

where $x_i \in R^n, y_i \in \{\pm 1\}, i = 1, \ldots, m$. The problem involves binary classification with m_1 positive samples and m_2 negative samples (where $m_1 + m_2 = m$), and the data is organized in matrix form. The positive class consists of data points represented by the matrix $A \in R^{m_1 \times n}$, where each row $A_i \in R^n$ corresponds to a sample of class A. Similarly, the negative class is represented by the matrix $B \in R^{m_2 \times n}$, where each row $B_i \in R^n$ corresponds to a sample of class B.

2.1. Parametric ν-support vector machine.

SVMs were originally developed for classification tasks, but their extension to regression, known as SVR, has gained significant attention. Standard SVR employs a loss function that ignores errors within a certain margin, known as the ϵ-insensitive zone, making it effective in handling noise. ν-SVR, an extension of traditional SVR, introduces the ν parameter, which provides control over the number of support vectors and the margin width. This flexibility allows ν-SVR to adapt to varying data distributions and improve generalization. Par-ν-SVR builds upon ν-SVR by introducing an additional set of parameters to optimize model flexibility and sparsity. By fine-tuning these parameters, Par-ν-SVR can effectively balance bias and variance, leading to better performance on heterogeneous data sets. The primary advantage of Par-ν-SVR is its adaptability in different

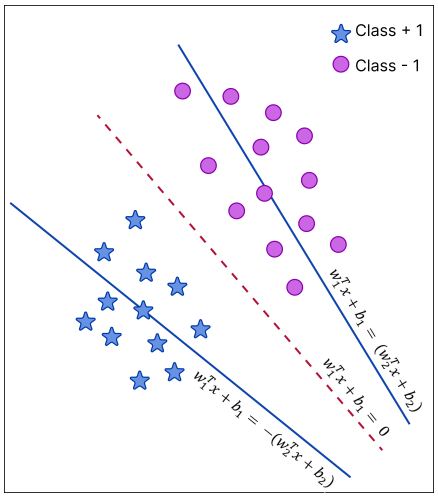

FIGURE 1. Par-ν-SVR.

regression scenarios, where standard SVR models might struggle due to rigid hyper parameters. By introducing a parametric-insensitive function $f^*(x)$, a new parametric-insensitive loss function $l_{f^*}(f,x,y)$ is derived as follows:

$$l_f^*(x,y,f) = (|y - f(x)| - f^*(x))_+,$$

where $f^*(x)$ is a real-valued function. Based on this, Hao [12] proposed Par-ν-SVR for estimating functions of the form:

$$f(x) = {w_1}^\top x + b_1, \quad f^*(x) = {w_2}^\top x + b_2.$$

The goal is to find the values of w_1, w_2, b_1, and b_2 that minimize the regularized risk functional:

$$\tfrac{1}{2} w_1^\top w_1 + C R_{emp}^{f^*(x)}[f],$$

where $R_{emp}^{f^*(x)}[f] = \frac{1}{m}\sum_{i=1}^m (|y_i - f(x_i)| - f^*(x_i))_+$ represents the $f^*(x)$-insensitive training error, and C is a constant that determines the trade-off. Minimizing this expression is equivalent to solving the constrained quadratic problem:

$$\begin{aligned} \min_{w_1,w_2,b_1,b_2,\xi_1,\xi_2} \quad & \frac{1}{2} w_1^\top w_1 + C\Big(\nu\Big(\frac{1}{2} w_2^\top w_2 + b_2\Big) + \frac{1}{m}(e_1^\top \xi_1 + e_2^\top \xi_2)\Big) \\ s.t. \quad & (Aw_1 + e_1 b_1) + (Aw_2 + e_1 b_2) \geq Y_1 - \xi_1, \\ & (Bw_2 + e_2 b_2) - (Bw_1 + e_2 b_1) \leq Y_2 + \xi_2, \\ & \xi_1, \xi_2 \geq 0, \end{aligned} \tag{2.1}$$

here, C and ν are positive penalty parameters. A graphical representation of Par-ν-SVR is shown in Figure 1. The Lagrangian function for this problem is:

$$\begin{aligned} L(w_1,w_2,b_1,b_2,\widetilde{\alpha_1},\widetilde{\alpha_2},\widetilde{\beta_1},\widetilde{\beta_2},\xi_1,\xi_2) = {} & \frac{1}{2} w_1^\top w_1 + C\Big(\nu(\frac{1}{2} w_2^\top w_2 + b_2) + \frac{1}{m}(e_1^\top \xi_1 + e_2^\top \xi_2)\Big) \\ & - \widetilde{\alpha_1}^\top [(Aw_1 + e_1 b_1) + (Aw_2 + e_1 b_2) - Y_1 + \xi_1] \\ & - \widetilde{\alpha_2}^\top [Y_2 + \xi_2 + (Bw_1 + e_2 b_1) - (Bw_2 + e_2 b_2)] \\ & - \widetilde{\beta_1}^\top \xi_1 - \widetilde{\beta_2}^\top \xi_2, \end{aligned}$$

$\widetilde{\alpha_1}, \widetilde{\alpha_2}, \widetilde{\beta_1}$, and $\widetilde{\beta_2}$ are non-negative Lagrange multipliers. Applying the Karush-Kuhn-Tucker (KKT) conditions, the dual optimization problem is obtained as follows:

$$\begin{aligned} \max_{\widetilde{\alpha_1},\widetilde{\alpha_2}} \ & -\frac{1}{2}(A^\top\widetilde{\alpha_1}+B^\top\widetilde{\alpha_2})^\top(A^\top\widetilde{\alpha_1}+B^\top\widetilde{\alpha_2})-\frac{1}{2C\nu}(A^\top\widetilde{\alpha_1}-B^\top\widetilde{\alpha_2})^\top(A^\top\widetilde{\alpha_1}-B^\top\widetilde{\alpha_2})+Y_1^\top\widetilde{\alpha_1}+Y_2^\top\widetilde{\alpha_2} \\ s.t. \ & -e_1^\top\widetilde{\alpha_1}-e_2^\top\widetilde{\alpha_2}=0, \\ & e_1^\top\widetilde{\alpha_1}-e_2^\top\widetilde{\alpha_2}=C\nu, \\ & 0\le\widetilde{\alpha_1}\le\frac{C}{m}e_1, \\ & 0\le\widetilde{\alpha_2}\le\frac{C}{m}e_2. \end{aligned} \tag{2.2}$$

Solving this dual problem yields the Lagrange multipliers $\widetilde{\alpha_1}$ and $\widetilde{\alpha_2}$, which allow us to compute the weight vectors w_1 and w_2:

$$\begin{aligned} w_1 &= A^\top\widetilde{\alpha_1}+B^\top\widetilde{\alpha_2}, \\ w_2 &= \frac{1}{C\nu}(A^\top\widetilde{\alpha_1}-B^\top\widetilde{\alpha_2}). \end{aligned} \tag{2.3}$$

The bias terms b_1 and b_2 can be determined using the KKT conditions:

$$\begin{aligned} b_1 &= -\frac{1}{2}(A_iw_1+B_iw_1+A_iw_2-B_iw_2), \\ b_2 &= -\frac{1}{2}(A_iw_1+A_iw_2-B_iw_1+B_iw_2), \end{aligned} \tag{2.4}$$

for $i=1,\dots,m$ and $\widetilde{\alpha_1},\widetilde{\alpha_2}\in(0,\frac{C}{m})$. The functions $f(x)$ and $f^*(x)$, the regression function and its corresponding parametric-insensitive function, can be obtained using (2.3) and (2.4). The following decision function predicts the class of a new sample $x_i\in R^n$:

$$f(x_i)=\begin{cases}+1 & \text{if } w_1^\top x_i+b_1>0,\\ -1 & \text{if } w_1^\top x_i+b_1<0.\end{cases}$$

2.2. **Parametric ν-support vector regression with Universum data.** Universum data consists of non-target samples that do not belong to either the training or test distribution but share structural properties with the target domain. The integration of Universum data into SVR aims to enhance generalization by regularizing the model and preventing over-fitting to specific patterns. By incorporating Universum data into Par-ν-SVR, the model leverages additional information that improves decision boundaries and robustness.

Consider a training data set $\widetilde{\chi}$ that is divided into two subsets:

$$\widetilde{\chi}=\chi\cup U,$$

where

$$\chi=\Big\{(x_1,y_1),\dots,(x_m,y_m)\Big\}\in(R^n\times\{\pm1\})^m, \text{ and } U=\{x_1^*,\dots,x_u^*\},$$

with $x_i\in R^n$, $y_i\in\{\pm1\}$ and $i=1,\dots,m$. The matrix $U\in R^{u\times n}$ represents the Universum class, where each row corresponds to a Universum sample. The objective is to enhance Par-ν-SVR by integrating Universum data, leading to the proposed model: UPar-ν-SVR. To classify data in UPar-ν-SVR, we use a hyperplane function:

$$f(x)=w_1^\top x+b_1,$$

unlike Par-ν-SVR, which relies on a functional margin, UPar-ν-SVR adopts a parametric margin model:

$$f^*(x)=w_2^\top x+b_2.$$

Thus, the decision function $f(x)=w_1^\top x+b_1$, classifies the data set if and only if:

$$\begin{aligned} (Aw_1+e_1b_1)&\ge(Aw_2+e_1b_2),\\ (Bw_1+e_2b_1)&\le-(Bw_2+e_2b_2). \end{aligned}$$

The geometric interpretation of UPar-ν-SVR is depicted in Figure 2. Consequently, the optimization problem for

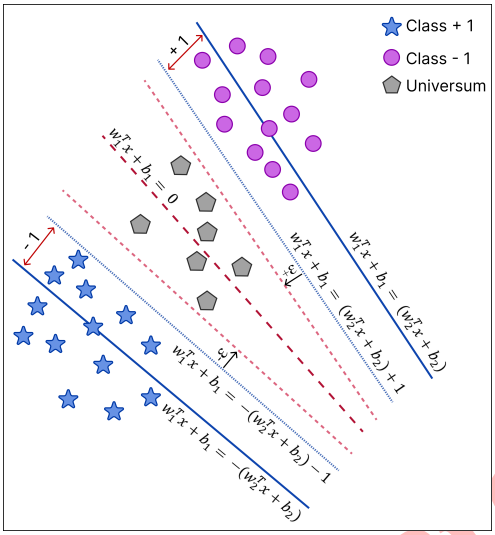

FIGURE 2. UPar-ν-SVR.

UPar-ν-SVR is formulated as:

$$\begin{aligned}
\min_{w_1,w_2,b_1,b_2,\xi_1,\xi_2,\psi_1,\psi_2} \quad & \frac{1}{2}w_1^\top w_1 + C\nu\Big(\frac{1}{2}w_2^\top w_2 + b_2\Big) + \frac{C^*}{m}(e_1^\top \xi_1 + e_2^\top \xi_2) + \frac{C_u^*}{u}(e_u^\top \psi_1 + e_u^\top \psi_2) \\
s.t. \quad & (Aw_1 + e_1 b_1) + (Aw_2 + e_1 b_2) \geq Y_1 - \xi_1, \\
& (Bw_2 + e_2 b_2) - (Bw_1 + e_2 b_1) \leq Y_2 + \xi_2, \\
& (Uw_1 + e_u b_1) + (Uw_2 + e_u b_2) \leq -e_u(1+\varepsilon) + \psi_1, \\
& (Uw_1 + e_u b_1) - (Uw_2 + e_u b_2) \geq e_u(1+\varepsilon) - \psi_2, \\
& \xi_1, \xi_2, \psi_1, \psi_2 \geq 0.
\end{aligned} \tag{2.5}$$

By solving this optimization problem, the parametric functions $f(x)$ and $f^*(x)$ can be determined. The corresponding Lagrangian function is:

$$\begin{aligned}
L(w_1, w_2, b_1, b_2, &\widetilde{\alpha_1}, \widetilde{\alpha_2}, \widetilde{\alpha_3}, \widetilde{\alpha_4}, \widetilde{\beta_1}, \widetilde{\beta_2}, \widetilde{\beta_3}, \widetilde{\beta_4}, \xi_1, \xi_2, \psi_1, \psi_2) \\
&= \frac{1}{2}w_1^\top w_1 + C\Big(\nu\Big(\frac{1}{2}w_2^\top w_2 + d\Big)\Big) \\
&+ \frac{C}{m}(e_1^\top \xi_1 + e_2^\top \xi_2) + \frac{C_u}{u}(e_u^\top \psi_1 + e_u^\top \psi_2) \\
&- \widetilde{\alpha_1}^\top [(Aw_1 + e_1 b_1) + (Aw_1 + e_1 b_2) - Y_1 + \xi_1] \\
&- \widetilde{\alpha_2}^\top [Y_2 + \xi_2 + (Bw2 + e_2 b_2) - (Bw_2 + e_2 b_2)] \\
&- \widetilde{\alpha_3}^\top [-e_u(1+\varepsilon) + \psi_1 - (Uw_1 + e_u b_1) - (Uw_2 + e_u b_2)] \\
&- \widetilde{\alpha_4}^\top [(Uw_1 + e_u b_1) - (Uw_2 + e_u b_2) - e_u(1+\varepsilon) + \psi_2] \\
&- \widetilde{\beta_1}^\top \xi_1 - \widetilde{\beta_2}^\top \xi_2 - \widetilde{\beta_3}^\top \psi_1 - \widetilde{\beta_4}^\top \psi_2,
\end{aligned}$$

here, $\widetilde{\alpha_1},\ldots,\widetilde{\alpha_4}$ and $\widetilde{\beta_1},\ldots,\widetilde{\beta_4}$ are non-negative Lagrange multipliers. The dual formulation is:

$$\begin{aligned}&\max\ L(\theta_1)\\ &s.t.\ \bigtriangledown_{\theta_2} L(\theta_1)=0,\ \theta_3\geq 0,\end{aligned}$$

where $\theta_1=(w_1,w_2,b_1,b_2,\widetilde{\alpha_1},\widetilde{\alpha_2},\widetilde{\alpha_3},\widetilde{\alpha_4},\widetilde{\beta_1},\widetilde{\beta_2},\widetilde{\beta_3},\widetilde{\beta_4},\xi_1,\xi_2,\psi_1,\psi_2)$,
$\theta_2=(w_1,w_2,b_1,b_2,\xi_1,\xi_2,\psi_1,\psi_2)$ and $\theta_3=(\widetilde{\alpha_1},\widetilde{\alpha_2},\widetilde{\alpha_3},\widetilde{\alpha_4},\widetilde{\beta_1},\widetilde{\beta_2},\widetilde{\beta_3},\widetilde{\beta_4})$.

By using KKT condition, we derive:

$$\begin{aligned}\max_{\widetilde{\alpha_1},\widetilde{\alpha_2},\widetilde{\alpha_3},\widetilde{\alpha_4}}\ & -\frac{1}{2}(A^\top\widetilde{\alpha_1}+B^\top\widetilde{\alpha_2}-U^\top\widetilde{\alpha_3}+U^\top\widetilde{\alpha_4})^\top(A^\top\widetilde{\alpha_1}+B^\top\widetilde{\alpha_2}-U^\top\widetilde{\alpha_3}+U^\top\widetilde{\alpha_4})\\ & -\frac{1}{2C\nu}(A^\top\widetilde{\alpha_1}-B^\top\widetilde{\alpha_2}+U^\top\widetilde{\alpha_3}-U^\top\alpha_4)^\top(A^\top\widetilde{\alpha_1}-B^\top\widetilde{\alpha_2}+U^\top\widetilde{\alpha_3}-U^\top\widetilde{\alpha_4})\\ & +Y_1^\top\widetilde{\alpha_1}+Y_2^\top\widetilde{\alpha_2}+e_u^\top(1+\varepsilon)\widetilde{\alpha_3}+e_u^\top(1+\varepsilon)\widetilde{\alpha_4},\\ s.t.\ & -e_1^\top\widetilde{\alpha_1}-e_2^\top\widetilde{\alpha_2}+e_u^\top\widetilde{\alpha_3}-e_u^\top\widetilde{\alpha_4}=0,\\ & {e_1}^\top\widetilde{\alpha_1}-e_2^\top\widetilde{\alpha_2}-e_u^\top\widetilde{\alpha_3}-e_u^\top\widetilde{\alpha_4}=C\nu,\\ & 0\leq\widetilde{\alpha_1}\leq\frac{C}{m}e_1,\\ & 0\leq\widetilde{\alpha_2}\leq\frac{C}{m}e_2,\\ & 0\leq\widetilde{\alpha_3}\leq\frac{C_u}{u}e_u,\\ & 0\leq\widetilde{\alpha_4}\leq\frac{C_u}{u}e_u.\end{aligned}\tag{2.6}$$

Solving these conditions yields:

$$w_1=A^\top\widetilde{\alpha_1}+B^\top\widetilde{\alpha_2}-U^\top\widetilde{\alpha_3}+U^\top\widetilde{\alpha_4},\tag{2.7}$$

$$w_2=\frac{1}{C\nu}(A^\top\widetilde{\alpha_1}-B^\top\widetilde{\alpha_2}+U^\top\widetilde{\alpha_3}-U^\top\widetilde{\alpha_4}).\tag{2.8}$$

$$b_1=-\frac{1}{2}(A_iw_1+B_iw_1+A_iw_2-B_iw_2),\tag{2.9}$$

$$b_2=-\frac{1}{2}(A_iw_1+A_iw_2-B_iw_1+B_iw_2).\tag{2.10}$$

Therefore, $f(x)=w_1^\top x+b_1$ and $f^*(x)=w_2^\top x+b_2$ are found. A new data point $x\in R^n$ is allocated to class $i\in\{+1,1\}$ by a similar rule to the Par-ν-SVR.

3. Modified Newton's method for solving parametric ν-support vector regression with Universum data

In this section, we present a novel method for addressing the problem (2.5) in the primal space. Our approach involves adapting Newton's method to solve this problem, and we call our technique NU par-ν-SVR. Subsequently, we

add 2-norm slack variables, denoted as ξ_1, ξ_2, ψ_1, and ψ_2, into the objective function of (2.5). Therefore, we obtain:

$$\begin{aligned}\min_{w_1,w_2,b_1,b_2,\xi_1,\xi_2,\psi_1,\psi_2} \ & \frac{1}{2}w_1^\top w_1 + C\nu\Big(\frac{1}{2}w_2^\top w_2 + b_2\Big) + \frac{C^*}{m}(\xi_1^\top \xi_1 + \xi_2^\top \xi_2) + \frac{C_u^*}{u}(\psi_1^\top \psi_1 + \psi_2^\top \psi_2) \\ s.t. \ & (Aw_1 + e_1 b_1) + (Aw_2 + e_1 b_2) \geq Y_1 - \xi_1, \\ & (Bw_2 + e_2 b_2) - (Bw_1 + e_2 b_1) \leq Y_2 + \xi_2, \\ & (Uw_1 + e_u b_1) + (Uw_2 + e_u b_2) \leq -e_u(1+\varepsilon) + \psi_1, \\ & (Uw_1 + e_u b_1) - (Uw_2 + e_u b_2) \geq e_u(1+\varepsilon) - \psi_2, \\ & \xi_1, \xi_2, \psi_1, \psi_2 \geq 0.\end{aligned} \tag{3.1}$$

In the optimal solution to problem (3.1), we obtain

$$\xi_1 = (Y_1 - (Aw_1 + e_1 b_1) + (Aw_2 + e_1 b_2))_+, \tag{3.2}$$

$$\xi_2 = ((Bw_2 + e_2 b_2) - (Bw_1 + e_2 b_1) - Y_2)_+, \tag{3.3}$$

$$\psi_1 = ((Uw_1 + e_u b_1) + (Uw_2 + e_u b_2) + e_u(1+\varepsilon))_+, \tag{3.4}$$

$$\psi_2 = (e_u(1+\varepsilon) - (Uw_1 + e_u b_1) + (Uw_2 + e_u b_2))_+. \tag{3.5}$$

In this case, $(.)_+$ sets any negative elements of a vector to zero. Consequently, we can substitute ξ_1, ξ_2, ψ_1, and ψ_2 in problem (3.1) with (3.2)-(3.5), transforming problem (3.1) into an equivalent form (3.6), which becomes an unconstrained optimization problem expressed as:

$$\begin{aligned}\min_{w_1,w_2,b_1,b_2} \ F(w_1,w_2,b_1,b_2) = & \frac{1}{2}w_1^\top w_1 + C\nu\Big(\frac{1}{2}w_2^\top w_2 + b_2\Big) + \frac{C^*}{m}(\|(Y_1 - (Aw_1 + e_1 b_1) \\ & + (Aw_2 + e_1 b_2))_+\|^2 + \|((Bw_2 + e_2 b_2) - (Bw_1 + e_2 b_1) - Y_2)_+\|^2) \\ & + \frac{C_u^*}{u}(\|((Uw_1 + e_u b_1) + (Uw_2 + e_u b_2) + e_u(1+\varepsilon))_+\|^2 \\ & + \|(e_u(1+\varepsilon) - (Uw_1 + e_u b_1) + (Uw_2 + e_u b_2))_+\|^2)\end{aligned} \tag{3.6}$$

The function $F(w_1, w_2, b_1, b_2)$ is piecewise quadratic, convex, and differentiable once. However, its gradient lacks differentiability. As a result, the objective function in problem (3.6) is not twice differentiable, preventing the use of the standard Newton's method to solve it. Hence, we begin by introducing the following concept:

Definition 3.1. If f is continuously differentiable on A and ∇f is locally Lipschitz on A, then f is a LC^1 function on an open set A.

This condition ensures that ∇f is differentiable almost everywhere within A, which allows us to define the generalized Hessian of f in the sense of Clarke [5]. Specifically, the generalized Hessian at a point x, denoted $\mathbb{H}_{f(x)}$, is the convex hull of all possible limits of Hessians at nearby differentiable points:

$$\mathbb{H}_{f(x)} = \text{conv}\big\{M \in \mathbb{R}^{n\times n} : \ \exists x_k \to x \text{ with } \nabla f \text{ differentiable at } x_k \text{ and } \nabla^2 f(x_k) \to M\big\},$$

where $conv(.)$ denotes the convex hull. The set $\mathbb{H}_{f(x)}$ forms a nonempty, compact, and convex subset of symmetric matrices. In the context of problem (3.6) , the second-order generalized derivative $\partial^2 \mathbf{F}(w_1, w_2, b_1, b_2)$ belongs to the set $\mathbb{H}_F$. Thus, we can apply the generalized Newton method to address the unconstrained optimization problem defined in (3.6). Since the generalized Hessian might be singular in some instances, we adopt a regularized Newton direction to maintain nonsingularity. This direction is computed as:

$$d_k = -(\partial^2 \mathbf{F}(\mathbf{x_k}) + \epsilon_0 I)^{-1} \nabla \mathbf{F}(x_k),$$

where $x_k = (w_{1k}, w_{2k}, b_{1k}, b_{2k})$ and ϵ_0 is a small positive constant (set to $\epsilon_0 = 10^{-6}$in our experiments), and I denotes the identity matrix of suitable size. The update rule for the generalized Newton algorithm is then given by:

$$x_{k+1} = x_k + \delta_k d_k,$$

where δ_k is a step size used at the k-th iteration. The procedure is initialized with an initial guess x_0, and the algorithm continues until convergence is achieved according to the following stopping condition:

$$\|x_{k+1} - x_k\| \leq tol,$$

with the tolerance level set to $tol = 10^{-8}$ in our numerical tests.

Thus, with the formulization above, the generalized Hessian can be defined. The problem can then be solved using the generalized Newton's approach (3.6).

Remark 3.2. (On time complexity) The generalized Newton method functions as an iterative procedure, and its computational cost depends on factors such as the algorithm's implementation details, the dimensionality of the problem, and the number of iterations required. Typically, for a problem involving n variables and k iterations, the time complexity can be expressed as $O(kn^3)$. Likewise, the linear NUPar-ν-SVR approach proceeds iteratively, where each cycle entails solving an unconstrained optimization problem (3.6). This problem includes $2n+2$ variables, resulting in an overall complexity of $O(k(2n+2)^3)$ after k iterations.

In the nonlinear case, problem (3.1) incorporates the following hyperplane:

$$K(x, D^\top)W + \lambda = 0,$$

where $D = [A^\top \quad B^\top]^\top$, $W \in R^n$, $\lambda \in R$ and $K(\cdot,\cdot)$ is an arbitrary kernel function. This leads to the formulation of the nonlinear optimization problem as follows:

$$\begin{aligned}
\min_{W_1,W_2,\lambda_1,\lambda_2,\xi_1,\xi_2,\psi_1,\psi_2} \ & \frac{1}{2}W_1^\top W_1 + C\nu\Big(\frac{1}{2}W_2^\top W_2 + \lambda_2\Big) + \frac{C^*}{m}(\xi_1^\top \xi_1 + \xi_2^\top \xi_2) + \frac{C_u^*}{u}(\psi_1^\top \psi_1 + \psi_2^\top \psi_2) \\
s.t. \ & (K(A, D^\top)W_1 + e_1\lambda_1) + (K(A, D^\top)W_2 + e_1\lambda_2) \geq Y_1 - \xi_1, \\
& (K(B, D^\top)W_2 + e_2\lambda_2) - (K(B, D^\top W_1 + e_2\lambda_1) \leq Y_2 + \xi_2, \\
& (K(U, D^\top)W_1 + e_u\lambda_1) + (K(U, D^\top)W_2 + e_u\lambda_2) \leq -e_u(1+\varepsilon) + \psi_1, \\
& (K(U, D^\top)W_1 + e_u\lambda_1) - (K(U, D^\top)W_2 + e_u\lambda_2) \geq e_u(1+\varepsilon) - \psi_2, \\
& \xi_1, \xi_2, \psi_1, \psi_2 \geq 0.
\end{aligned} \tag{3.7}$$

Following a similar procedure in the linear case, the unconstrained problem (3.7) can be addressed as follows:

$$\begin{aligned}
\min_{W_1,W_2,\lambda_1,\lambda_2} \widetilde{F}(W_1, W_2, \lambda_1, \lambda_2) = & \frac{1}{2}W_1^\top W_1 + C\nu\Big(\frac{1}{2}W_2^\top W_2 + b_2\Big) + \frac{C^*}{m}(\|(Y_1 - (K(A, D^\top)W_1 + e_1\lambda_1) \\
& + (K(A, D^\top)W_2 + e_1\lambda_2))_+\|^2 + \|((K(B, D^\top)W_2 + e_2\lambda_2) - (K(B, D^\top)W_1 + e_2\lambda_1) \\
& - Y_2)_+\|^2) + \frac{C_u^*}{u}(\|((K(U, D^\top)W_1 + e_u\lambda_1) + (K(U, D^\top)W_2 + e_u\lambda_2) + e_u(1+\varepsilon))_+\|^2 \\
& + \|(e_u(1+\varepsilon) - (K(U, D^\top)W_1 + e_u\lambda_1) + (K(U, D^\top)W_2 + e_u\lambda_2))_+\|^2)
\end{aligned} \tag{3.8}$$

After resolving the optimization problem mentioned above, we produce a non-parallel classifier by applying the generalized Newton's approach.

4. Numerical experiment

Several UCI data sets [8] are used in the numerical experiment section to compare the suggested approach (NUPar-ν-SVR) with Par-ν-SVRC$^+$ [17] and UPar-ν-SVR [26]. Using Matlab R2018b, all numerical experiments are conducted on a PC running Microsoft Windows 64-bit with a Core(TM) i7 CPU operating at 2.20 GHz and 4GB of RAM. Additionally, we applied the five-fold cross-validation technique to evaluate the algorithm's performance and classification accuracy. In each benchmark data set, we randomly selected an equal number of samples from each class. To generate the Universum data, we then averaged pairs of samples belonging to different classes, using half of the selected data. The experimental findings for linear and nonlinear states on several UCI data sets are compiled in Tables 1 and 2, where "Acc(%)" indicates the accuracy mean value plus or minus the associated standard deviation. "Time(s)" denotes the algorithm's execution time.

Choosing the right parameter values is important for getting good results from a machine learning model. Grid search is a common method used to find the best values by testing different combinations of parameters and selecting the one with the highest performance [14]. In our experiments, we used grid search to find the best set of parameters. As shown in Figure 3, different parameter values (such as C and γ) have a significant impact on the model's accuracy.

The effectiveness of these methods relies heavily on the selection of the parameters C, C^*, C_u^*, and ν. For Par-ν-SVR, UPar-ν-SVR, and NUPar-ν-SVR, the ideal values were determined as follows: C, C^*, and C_u^* were chosen from the set $\{2^i | i = 10, ..., 10\}$, while ν was selected from the range $\{0.1, ..., 0.9\}$. Additionally, a Gaussian kernel, defined as $K(x, y) = \exp(-\gamma \|x - y\|^2)$, $\gamma > 0$ where $\gamma > 0$, was utilized for all methods in their nonlinear forms. The value of the kernel parameter γ was picked from the set $\{2^i \mid i = -10, \ldots, 10\}$. The Gaussian kernel is commonly used in related studies because of its flexibility in capturing nonlinear relationships and its consistent empirical performance across a wide range of data sets, including those from the UCI benchmark data sets repository. As demonstrated by Tables 1 and 2, our suggested approach achieves the best classification accuracy for the majority of UCI data sets using both linear and nonlinear states. Although the proposed method generally achieves strong performance across most data sets, we observe a decrease in accuracy on certain data sets, such as the Housing data set in linear state. This may be attributed to the specific properties of the data. Moreover, in our approach, the Universum data are constructed by averaging randomly selected pairs of samples from different classes. While this method helps introduce neutral, non-target information into the learning process, its effectiveness may vary depending on how well the generated Universum points reflect the structure of the actual data space. Such variations can influence the model's ability to generalize, particularly in datasets with unique or sparse feature distributions. Therefore, we conclude that, compared to other approaches, our suggested strategy has the best classification accuracy and the most manageable learning time.

5. Conclusion

This study presented a novel method for resolving the UPar-ν-SVR optimization challenge. We modified the constrained quadratic programming problem (UPar-ν-SVR) into an unconstrained quadratic problem by utilizing the 2-norm of the slack vectors in the quadratic problem's objective functions. We then proposed a modified version of Newton's method for dealing with unconstrained quadratic situations, which simplifies and expedites the solution of the associated problem.

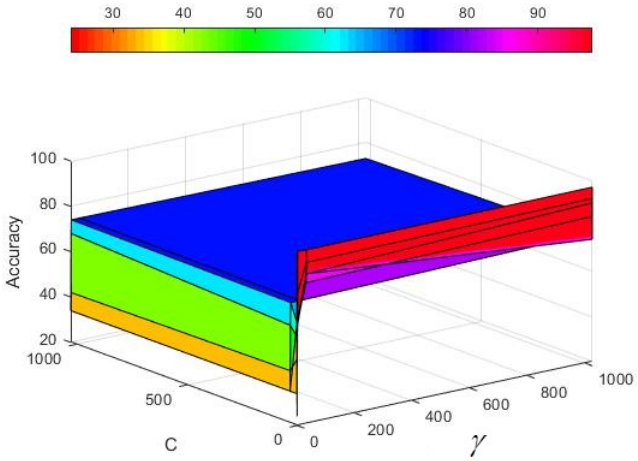

Figure 3. The impact of selecting different values for parameters γ and C on the accuracy of Cancer data sets in nonlinear state.

TABLE 1. Performance comparison of linear algorithm's on UCI data sets.

Data set Size	Par-ν-SVR Acc (%) Time (s)	Par-ν-SVRC$^+$ Acc (%) Time (s)	NUPar-ν-SVR Acc (%) Time (s)
German 1000×24	70 ±0.00 18.04	70±0.00 0.56	**77.20±2.79** 0.65
Ripley 250×2	67.60±0.21 1.36	**86.40±2.19** 0.27	**86.40±4.09** 0.39
Heart 270×16	79.62±3.92 3.07	85.55±4.22 0.56	**85.92±4.05** 0.67
F-diagnosis 100×9	**88.03±2.46** 1.40	**88.03±2.46** 0.28	86.04±4.03 0.29
Wdbc 569×30	82.97±3.17 2.13	88.25±5.25 0.42	**96.14±1.78** 0.44
House Vote 435×16	92.41±3.86 1.46	96.78±2.74 0.30	**97.01±2.88** 0.33
Housing 506×14	**93.28±4.36** 1.76	82.58±1.80 0.37	71.48±5.95 0.45
Haberman 306×3	**73.53±0.00** 1.77	**73.53±5.32** 0.34	68.17±5.81 0.43
Bupa 345×6	58.84±3.63 1.43	59.13±2.59 0.34	**70.14±6.60** 0.39
Splice 1000×60	71.39±3.59 9.05	73.29±2.97 0.88	**81.39±3.92** 1.22
Ionosphere 351×34	84.04±3.12 1.41	87.75±5.93 0.32	**88.31±6.34** 0.37
Sonar 208×60	61.54±5.24 1.34	77.87±3.23 0.31	**78.37±3.33** 0.33
Cancer 699×6	90.59±7.42 1.82	**97.42±8.27** 0.33	97.28±6.02 0.37

ACKNOWLEDGMENT

The research of F. Bazikar and B. Sadeghi Bigham has been facilitated by the Data Science Lab of the Faculty of Mathematical Sciences at Alzahra University.

TABLE 2. Performance comparison of nonlinear algorithm's on UCI data sets.

Data set Size	Par-ν-SVR Acc (%) Time (s)	Par-ν-SVRC$^+$ Acc (%) Time (s)	NUPar-ν-SVR Acc (%) Time (s)
German 1000×24	45.90 ±0.01 6.26	72.20±0.00 24.49	**73.40±3.28** 28.95
Ripley 250×2	86±0.11 1.42	88.80±1.23 0.80	**89.60±2.60** 0.87
Heart 270×16	81.11±1.51 4.37	84.81±1.27 1.18	**85.18±2.41** 1.26
F-diagnosis 100×9	89.03±0.21 1.54	89.12±0.33 0.32	**89.17±2.46** 0.48
Wdbc 569×30	45.25±2.19 2.19	63.44±0.55 5.28	**63.45±0.90** 6.86
House Vote 435×16	92.41±0.11 1.61	96.54±1.54 3.41	**96.79±2.03** 3.93
Housing 506×14	43.98± 0.24 6.86	**94.06±0.21** 5.65	**94.06±0.70** 5.66
Haberman 306×3	**75.49±0.00** 2.46	74.84±0.21 1.44	74.84±1.52 1.96
Bupa 345×6	49.24±0.61 2.47	68.98±0.00 1.47	**69.56±0.88** 1.51
Splice 1000×60	68.89±0.09 32.57	87.50±1.27 29.74	**88.39±0.98** 31.42
Ionosphere 351×34	91.44±0.21 1.92	89.12±0.32 1.52	**89.17±0.24** 1.86
Sonar 208×60	62.49±1.22 1.45	84.08±0.00 0.72	**90.39±4.39** 0.75
Cancer 699×6	94.55±0.33 8.59	96.57±1.11 7.23	**97.34±0.42** 8.31

References

[1] Z. Arabasadi, R. Alizadehsani, M. Roshanzamir, H. Moosaei, and A. A. Yarifard, *Computer aided decision making for heart disease detection using hybrid neural network-Genetic algorithm*, Computer Methods and Programs in Biomedicine, *141* (2017), 19–26.

[2] N. Ayoobi et al., *Time series forecasting of new cases and new deaths rate for COVID-19 using deep learning methods*, Results in Physics, *27* (2021), 104495.

[3] F. Bazikar, S. Ketabchi, and H. Moosaei, *DC programming and DCA for parametric-margin ν-support vector machine*, Applied Intelligence, *50* (2020), 1763–1774.

[4] O. Chapelle, A. Agarwal, F. Sinz, and B. Schölkopf, *An analysis of inference with the universum*, Advances in Neural Information Processing Systems, *20* (2007).

[5] F.H. Clarke, *Optimization and Nonsmooth Analysis*, SIAM, (1990).

[6] S. Ding, N. Zhang, X. Zhang, and F. Wu, *Twin support vector machine: theory, algorithm and applications*, Neural Computing and Applications, *28* (2017), 3119–3130.

[7] M. H. Doroudyan and S. T. A. Niaki, *Pattern recognition in financial surveillance with the ARMA-GARCH time series model using support vector machine*, Expert Systems with Applications, *182* (2021), 115334.

[8] D. Dua and C. Graff, *UCI machine learning repository*, (2019), URL https://archive.ics.uci.edu/ml.

[9] M. Ganaie, M. Tanveer, and J. Jangir, *EEG signal classification via pinball universum twin support vector machine*, Annals of Operations Research, *328*(1) (2023), 451–492.

[10] M. Ganaie, M. Tanveer, and A. s. D. N. Initiative, *KNN weighted reduced universum twin SVM for class imbalance learning*, Knowledge-Based Systems, *245*(2022), 108578.

[11] D. Gupta, H. J. Sarma, K. Mishra, and M. Prasad, *Regularized Universum twin support vector machine for classification of EEG Signal*, in 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), IEEE, (2019), 2298–2304.

[12] P.-Y. Hao, *New support vector algorithms with parametric insensitive/margin model*, Neural Networks, *23*(1) (2010), 60–73.

[13] B. Hazarika, D. Gupta, and B. Kumar, *EEG signal classification using a novel universum-based twin parametric-margin support vector machine*, Cognitive Computation, *16*(4) (2024), 2047–2062.

[14] C.W. Hsu, C.C. Chang, C.J. Lin, et al., *A practical guide to support vector classification*, (2003), URL: https://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf

[15] S. H. Javadi, H. Moosaei, and D. Ciuonzo, *Learning wireless sensor networks for source localization*, Sensors, *19*(3) (2019), 635.

[16] W. Jianlin, F. Xuying, and Y. Tao, *A geometric approach to support vector regression and its application to fermentation process fast modeling*, Chinese Journal of Chemical Engineering, *20*(4) (2012), 715–722.

[17] S. Ketabchi, H. Moosaei, M. Razzaghi, and P. M. Pardalos, *An improvement on parametric ν-support vector algorithm for classification*, Annals of Operations Research, *276*(1) (2019), 155–168.

[18] R. Khemchandani and S. Chandra, *Twin support vector machines for pattern classification*, IEEE Transactions on Pattern Analysis and Machine Intelligence, *29*(5) (2007), 905–910.

[19] M. A. Kumar and M. Gopal, *Least squares twin support vector machines for pattern classification*, Expert Systems with Applications, *36*(4) (2009), 7535–7543.

[20] B. Kumar and D. Gupta, *Universum based Lagrangian twin bounded support vector machine to classify EEG signals*, Computer Methods and Programs in Biomedicine, *208* (2021), 106244.

[21] B. Liu et al., *Adaptive robust Adaboost-based twin support vector machine with universum data*, Information Sciences, *609* (2022), 1334–1352.

[22] O. L. Mangasarian and E. W. Wild, *Multisurface proximal support vector machine classification via generalized eigenvalues*, IEEE Transactions on Pattern Analysis and Machine Intelligence, *28*(1) (2005), 69–74.

[23] B. Mei and Y. Xu, *Multi-task least squares twin support vector machine for classification*, Neurocomputing, *338* (2019), 26–33.

[24] H. Moosaei, S. Ketabchi, M. Razzaghi, and M. Tanveer, *Generalized twin support vector machines*, Neural Processing Letters, *53*(2) (2021), 1545–1564.

[25] H. Moosaei and M. Hladík, *A lagrangian-based approach for universum twin bounded support vector machine with its applications*, Annals of Mathematics and Artificial Intelligence, *91*(2) (2023), 109–131.
[26] H. Moosaei, F. Bazikar, and M. Hladík, *Universum parametric ν-support vector regression for binary classification problems with its applications*, Annals of Operations Research, (2023), 1–45.
[27] H. Moosaei, F. Bazikar, S. Ketabchi, and M. Hladík, *Universum parametric-margin ν-support vector machine for classification using the difference of convex functions algorithm*, Applied Intelligence, *52*(3) (2022), 2634–2654.
[28] H. Moosaei, M. Ganaie, M. Hladík, and M. Tanveer, *Inverse free reduced universum twin support vector machine for imbalanced data classification*, Neural Networks, *157* (2023), 125–135.
[29] W. S. Noble, *Support vector machine applications in computational biology*, in Kernel Methods in Computational Biology, (2004), 71.
[30] X. Peng, *Least squares twin support vector hypersphere (LS-TSVH) for pattern recognition*, Expert Systems with Applications, *37*(12) (2010), 8371–8378.
[31] X. Peng, *A ν-twin support vector machine (ν-TSVM) classifier and its geometric algorithms*, Information Sciences, *180*(20) (2010), 3863–3875.
[32] X. Peng, *TPMSVM: a novel twin parametric-margin support vector machine for pattern recognition*, Pattern Recognition, *44*(10–11) (2011), 2678–2692.
[33] Z. Qi, Y. Tian, and Y. Shi, *Twin support vector machine with universum data*, Neural Networks, *36* (2012), 112–119.
[34] B. Richhariya and M. Tanveer, *EEG signal classification using universum support vector machine*, Expert Systems with Applications, *106*(2018), 169–182.
[35] B. Richhariya, M. Tanveer, and A. S. D. N. Initiative, *A fuzzy universum least squares twin support vector machine (FULSTSVM)*, Neural Computing and Applications, *34*(14) (2022), 11411–11422.
[36] B. Richhariya and M. Tanveer, *Universum least squares twin parametric-margin support vector machine*, in 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, (2020), 1–8.
[37] B. Richhariya, M. Tanveer, and Alzheimer's Disease Neuroimaging Initiative, *An efficient angle-based universum least squares twin support vector machine for classification*, ACM Transactions on Internet Technology, *21*(3) (2021), 1–24.
[38] Y. H. Shao, C. H. Zhang, X. B. Wang, and N. Y. Deng, *Improvements on twin support vector machines*, IEEE Transactions on Neural Networks, *22*(6) (2011), 962–968.
[39] Y. H. Shao, Z. Wang, W. J. Chen, and N. Y. Deng, *Least squares twin parametric-margin support vector machine for classification*, Applied Intelligence, *39* (2013), 451–464.
[40] M. Tanveer, A. Sharma, and P. N. Suganthan, *General twin support vector machine with pinball loss function*, Information Sciences, *494* (2019), 311–327.
[41] M. Tanveer, A. Tiwari, R. Choudhary, and M. Ganaie, *Large-scale pinball twin support vector machines*, Machine Learning, (2022), 1–24.
[42] Y. Tian and Z. Qi, *Review on: twin support vector machines*, Annals of Data Science, *1* (2014), 253–277.
[43] Y. Tian, Q. Zhang, and D. Liu, *ν-Nonparallel support vector machine for pattern classification*, Neural Computing and Applications, *25* (2014), 1007–1020.
[44] Y. Tian, Z. Qi, X. Ju, Y. Shi, and X. Liu, *Nonparallel support vector machines for pattern classification*, IEEE Transactions on Cybernetics, *44*(7) (2013), 1067–1079.
[45] V. Vapnik, *The support vector method of function estimation*, in Nonlinear Modeling: Advanced Black-Box Techniques: Springer, (1998), 55–85.
[46] V. Vapnik and A. Chervonenkis, *Theory of pattern recognition*, Nauka, Moscow, 1974.
[47] J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik, *Inference with the universum*, in Proceedings of the 23rd International Conference on Machine Learning, (2006), 1009–1016.
[48] X. Y. Wang, T. Wang, and J. Bu, *Color image segmentation using pixel wise support vector machine classification*, Pattern Recognition, *44*(4) (2011), 777–787.
[49] Y. Xiao, J. Wen, and B. Liu, *A new multi-task learning method with universum data*, Applied Intelligence, *51* (2021), 3421–3434.

[50] Y. Xu, M. Chen, and G. Li, *Least squares twin support vector machine with universum data for classification*, International Journal of Systems Science, *47*(15) (2016), 3637–3645.
[51] Z. Yang and Y. Xu, *Laplacian twin parametric-margin support vector machine for semi-supervised classification*, Neurocomputing, *171* (2016), 325–334.
[52] H. Zamani Sabzi, S. Abudu, R. Alizadeh, L. Soltanisehat, N. Dilekli, and J. P. King, *Integration of time series forecasting in a dynamic decision support system for multiple reservoir management to conserve water sources*, Energy Sources, Part A: Recovery, Utilization, and Environmental Effects, *40*(11) (2018), 1398–1416.
[53] J. Zhao, Y. Xu, and H. Fujita, *An improved non-parallel universum support vector machine and its safe sample screening rule*, Knowledge-Based Systems, *170* (2019), 79–88.