Applied Mathematics and Computation 510 (2026) 129705

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Full Length Article

Gravitational least squares twin support vector machine based on optimal angle for class imbalance learning

Abdullah Mohammadi [a], Jalal A. Nasiri [b,*], Sohrab Effati [a,c]

[a] Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, P. O. Box 1159, Mashhad, 91775, Iran
[b] Department of Computer Sciences, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, P. O. Box 1159, Mashhad, 91775, Iran
[c] Center of Excellence of Soft Computing and Intellegent Information Processing(SCIIP), Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Keywords:
Twin support vector machine
Optimal angle
Gravitational LSTSVM
Class imbalance learning
Large scale data

ABSTRACT

This paper introduces the Gravitational Least Squares Twin Support Vector Machine for Class Imbalance Learning (GLSTSVM-CIL), a novel binary classification method designed to address critical limitations in existing approaches for imbalanced large-scale datasets. Traditional methods like Fuzzy TSVM and KNN-based weighting fail to simultaneously capture both global positional relationships and local density characteristics of data points. Our proposed gravitational weighting function innovatively models data samples as masses influenced by their distance from class centroids and neighborhood density, effectively prioritizing representative points while suppressing outliers. The optimization framework uniquely incorporates angular constraints between hyperplanes to enhance structural risk control and generalization capability. For scalability, we reformulate the solution into a linear system solvable via conjugate gradient methods, avoiding computationally expensive matrix inversions. Comprehensive evaluations on 92 datasets (including synthetic, noisy, medical, text, and large-scale NDC benchmarks) demonstrate GLSTSVM-CIL's superior performance, particularly in minority-class recognition where it achieves average F1-Score improvements over baseline methods. The model maintains robust Accuracy under high noise (20 %) and extreme class imbalance (ratio 20:1) while ables to process datasets up to 50,000 samples.

1. Introduction

In binary classification problems-where data are labeled as belonging to either a positive or negative class-Support Vector Machine (SVM) has proven to be a powerful and well-established method [1]. SVM has been successfully applied in various domains including disease diagnosis, text classification, speech and facial recognition, bankruptcy prediction, intrusion detection, and sentiment analysis [2]. Despite its success, SVM faces several limitations: its training involves solving a large quadratic optimization problem, which becomes computationally expensive for high-dimensional data; and the assumption of a single optimal separating hyperplane can be overly rigid for many real-world datasets.

To address these concerns, Twin Support Vector Machine (TSVM) was introduced by Jayadeva et al. [3], which reformulates the SVM optimization into two smaller Quadratic Programming Problems (QPPs), each favoring the respective class. TSVM offers better efficiency and improved classification Accuracy in many applications. Building upon this, Least Squares TSVM (LSTSVM)

* Corresponding author.
E-mail addresses: abdhmohammadi@alumni.um.ac.ir (A. Mohammadi), JNasiri@um.ac.ir (J.A. Nasiri), s-effati@um.ac.ir (S. Effati).

https://doi.org/10.1016/j.amc.2025.129705
Received 24 December 2024; Received in revised form 24 June 2025; Accepted 26 August 2025
Available online 4 September 2025

was proposed by Kumar and Gopal [4], converting the QPPs into two linear equations, making the method faster and simpler by eliminating the need for external optimization tools.

Further improvements have been made to the TSVM framework. LS-ATWSVM incorporates angular optimization between hyperplanes and has demonstrated superior performance in both Accuracy and training time [5]. However, these models often treat all data points equally, ignoring the fact that not all samples contribute equally to classification boundaries.

In the mentioned methods, all the data play the same rule in determining the hyperplanes. But in the real world, this is not always true. To take advantage of this fact, weighting methods were presented in the field of support vector machines. In Fuzzy Twin Support Vector Machine (FTSVM), a membership function is used to weight the data, which produces membership values in the (0,1). This function is defined based on the distance from the center of the class. Using this approach, different weights are given to the points, which results in less influence from outliers and noise [6]. However, FTSVM does not incorporate local data density. To integrate neighborhood information, [7] proposed KNN-LSTSVM, which leverages k-nearest neighbor density to assign weights and capture local structure.

Despite these advancements, many methods still struggle when applied to imbalanced data-a common scenario where one class significantly outnumbers the other. This imbalance can cause the learned decision boundary to skew towards the majority class, leading to poor classification of the minority class, which is often the class of higher interest (e.g., in medical diagnosis such as COVID-19 detection). Recent studies have proposed advanced hybrid methods to mitigate these issues. One such approach combines SVM with an improved Simulated Annealing (SA) algorithm [8]. This technique first applies data preprocessing to balance class distributions through synthetic sample generation, followed by data reduction to eliminate redundancy. The improved SA algorithm optimizes the penalty parameter in SVM, enhancing classification performance. Experimental validation showed that this approach outperforms conventional SVM implementations and effectively combats majority-class error. Another innovative strategy is the Adaptive SV-Borderline SMOTE-SVM Algorithm, which enhances minority-class representation by strategically generating synthetic samples [9]. Instead of applying synthetic sample generation broadly, this method selects support vectors (SVs) as borderline sampleskey points influencing the decision boundary. Using adaptive sample generation in kernel space, it ensures that the new instances align optimally with the classifier's structure. This integration of Borderline-SMOTE with SVM improves minority-class learning, reduces bias toward majority classes, and strengthens overall model generalization.

To further advance imbalanced data classification, we propose a new model, *Gravitational Least Squares Twin Support Vector Machine for Class Imbalance Learning (GLSTSVM-CIL)*, inspired by physical gravitational theory and recent TSVM variants such as LSFLSTSVM-CIL [10] and LS-ATWSVM [5]. This method combines the influence of both distance and density through a gravity-inspired weight function and includes several improvements:

1. Each data point is treated as a "mass" influenced by its surrounding neighborhood and its distance from the class center, capturing both local density and global position.
2. The gravitational weight function reflects the force of attraction toward the class center, analogously balancing distance and local accumulation.
3. An additional angular optimization term improves structural risk control, enhancing generalization.
4. The weighting mechanism is applied to both majority and minority classes, improving balanced detection performance.
5. The model avoids matrix inversion by using numerically efficient solvers, improving scalability.

The rest of this paper is organized as follows: Section 2 surveys related works. Section 3 introduces Proposed Method in detail. Section 4 analyzes the Solution method, Computational complexity and Convergence of the solution method, Section 5 presents experimental evaluations, including statistical analysis. And Section 6 presents real-world applications in text and medical classification.

2. Related works

Learning from imbalanced datasets is a major challenge in many real-world applications such as fraud detection, medical diagnosis, and text classification. Traditional classifiers often bias towards the majority class, leading to poor performance on the minority class. Many efforts have been made to tackle this issue, such as cost-sensitive learning [11], data-level resampling techniques, ensemble-based methods [12], and margin-based classifiers including SVM variants. Twin Support Vector Machines (TSVM) introduced by Khemchandani and Jayadeva [3] became a notable alternative to classical SVM by learning two non-parallel hyperplanes, one for each class. Various extensions have been proposed to improve TSVM performance under imbalanced settings. To improve robustness against noisy data, fuzzy extensions such as FLSTSVM [13] and its kernelized versions have been developed.

More recently, Richhariya and Singh [14] proposed the RFLSTSVM-CIL, which incorporates class-specific fuzzy membership and imbalance-aware regularization. Ganaie and Hu [15] reviewed modern ensemble and hybrid methods for imbalanced learning, emphasizing the need for structure-aware and scalable classifiers.

Suppose the dataset D with m samples is defined as follow:

$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^n, y_i \in \{1, -1\}, i = 1, \dots, m\}, \tag{1}$$

which every x_i refers a sample and y_i is label of x_i. Also, the subset $A = \{x_i | (x_i, y_i) \in D, y_i = +1\}$ as positive class samples and $B = \{x_i | (x_i, y_i) \in D, y_i = -1\}$ to be considered as a subset of the negative class samples (In this paper, without losing the generality of the problem, the smaller class is considered as the positive samples) such that $A \cup B = D$ and $A \cap B = \emptyset$. The goal is to find two

hyperplanes as follows:

$$\begin{cases} w_1^T x + b_1 = 0, \\ w_2^T x + b_2 = 0 \end{cases} \tag{2}$$

where $b_1, b_2 \in \mathbb{R}$ and $w_1, w_2 \in \mathbb{R}^n$ are the parameters of hyperplanes.

2.1. Angle based least squares TSVM(LS-ATWSVM)

Angle-based TSVMs, focus on maximizing the angular separation between hyperplanes to increase discriminative power. LS-ATWSVMis based on weightless TSVM concept, whose main goal is to maximize the angle between two hyperplanes [5]. The first LS-ATWSVM problem with the modifications made on the LSTSVM model was written as follows:

$$\underset{w_1,b_1}{min} \frac{c_1}{2}(\|w_1\|^2 + b_1^2) + \frac{1}{2}(Aw_1^T + eb_1)^2 + \frac{c_3}{2}\|Bw_1^T + eb_1 + e\|^2. \tag{3}$$

After setting the gradient of Eq. (3) with respect to w_1 and b_1 equal to zero, it is solved as follows:

$$u_1 = -c_3\left(c_1 I_{n+1} + H^T H + c_3 G^T G\right)^{-1} G^T e, \tag{4}$$

where $H = [A\ e]$ and $G = [B\ e]$ are the augmented matrices of A and B. The I_{n+1} matrix is added to avoid encountering singulare matrices. After solving w_1 and b_1, these values are used as norm vector of hyperplane to maximize the angle in the second problem. The second problem is as follows:

$$\underset{w_2,b_2}{min} \frac{c_5}{2}(\|w_2\|^2 + b_2^2) + c_2\|Bw_2^T + eb_2 + e\|^2 + c_4(w_1 w_2^T + b_1 b_2). \tag{5}$$

Setting the gradient of Eq. (5) with respect to w_2 and b_2 equal to zero, is solved as follows:

$$u_2 = -\frac{c_4}{2c_2}\left(G^T G + \frac{c_5}{2c_2} I_{n+1}\right)^{-1} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix}. \tag{6}$$

The class of the new sample x is specified according to the following pattern:

$$class(x) = \underset{i=1,2}{argmin} \frac{|w_i x^T + b_i|}{\|w_i\|^2}. \tag{7}$$

2.2. Large scale fuzzy LSTSVM for class imbalance learning

Nonlinear version of LSFLSTSVM-CIL is described here, First, based on [14,16,17], the class imbalance ratio is defined as follows:

$$IR = \frac{\text{number of data points in B}}{\text{number of data points in A}} = \frac{N(B)}{N(A)}. \tag{8}$$

Then the following membership function is used:

$$W = \frac{1}{1+IR} + \left(\frac{IR}{1+IR}\right)\left(\frac{e^{c_0\left(\frac{d_1(x)-d_2(x)}{d} - \frac{d_2}{R_2}\right)} - e^{-2c_0}}{e^{c_0} - e^{-2c_0}}\right), \tag{9}$$

where $d_1(x)$ and $d_2(x)$ are the Euclidean distance of the x from the center of the positive class and the negative class, respectively. d is the Euclidean distance between the center of two classes. R_2 is the maximum distance of negative samples from its center and c_0 is a parameter for exponential scaling. To deal with non-linear data, kernel functions are used, Kernel functions are functions that transfer data to a higher space using dot Product, which usually makes it possible to separate points linearly [18]. These functions are generally defined as follows:

$$K(x, y) = \phi^T(x)\phi(y). \tag{10}$$

Optimization problem of the LS-FLSTSVM-CIL is constructed by adding a term to the objective function of RFLSTSVM-CIL [14] as follows:

$$\begin{aligned} &min \frac{c_3}{2}(\|w_1\|^2 + b_1^2) + \frac{1}{2}\eta_1^T\eta_1 + \frac{c_1}{2}(S_2\xi_2)^T(S_2\xi_2) \\ &S.t.\ \ \phi(A)w_1^T + eb_1 = \eta_1, \\ &-(\phi(B)w_1^T + eb_1) + \xi_2 = e, \end{aligned} \tag{11}$$

and

$$min \frac{c_4}{2}(\|w_2\|^2 + b_2^2) + \frac{1}{2}\eta_2^T\eta_2 + \frac{c_2}{2}(S_1\xi_1)^T(S_1\xi_1)$$

$$S.t.\ \ \phi(B)w_2^T + eb_2 = \eta_2,$$
$$-(\phi(A)w_2^T + eb_2) + \xi_1 = e. \tag{12}$$

The first term in the objective functions implements the principle of structural risk. The second term is the sum of the squares of the distance between the points of each class of the corresponding hyperplane and brings the hyperplane closer to the points of the same class. The third term also penalizes samples based on their distance from the center of the class through fuzzy membership weights S_1 and S_2. The constraint corresponding to the second term of the objective function is to minimize the distance of the points of each class from its hyperplane, and the second constraint ensures that the points of the other class are at a distance of one unit from the hyperplane. The confusion of this interval is given by the ξ_i, which is penalized in the objective function using fuzzy membership weights.

3. Proposed method: GLSTSVM-CIL

In this section, Proposed Method titled *"Gravitational Least Squares Twin Support Vector Machine for Class Imbalance Learning"*, which is abbreviated as *"GLSTSVM-CIL"*, is described.

3.1. Weight function

As discussed earlier, both the distance of data points from the class center and the local density around them play critical roles in estimating the importance or representativeness of the samples. A data point that is either centrally located or surrounded by many neighboring points is intuitively more significant than isolated or distant ones. The combination of these two factors-density and distance-motivated us to design a novel weighting function that is inspired by the structure of Newtonian[1] gravitational force. This gravitational perspective is a metaphorical and heuristic tool, similar to how gravitational principles have been adapted in various data-driven contexts such as swarm intelligence and clustering [19,20].

The gravitational force between two masses m_1 and m_2 separated by a distance r in physics is given by [21]:

$$F = G\frac{m_1 \times m_2}{r^2}, \tag{13}$$

where G is the universal gravitational constant. Inspired by this formulation, we define a data-driven weight function for each sample based on a combination of its distance from the class center and its local density, which we interpret as a proxy for "mass".

In this analogy, within each class m_1 represents the "mass" of the class center, m_2 is the "mass" of a sample point, computed based on its local density (detailed in Section 3.1.1) and r is the distance between the sample and the class center in the feature space. This metaphor helps justify why denser and more central points should receive larger weights, while outliers-those that are far from the center and lie in sparse regionsshould contribute less. Fig. 1 illustrates this idea:

- Point p_1 is far from the class center but lies in a dense region, hence receives a moderate weight.
- Point p_2 is near the class center but lies in a sparse region, also justifying a reasonable weight.
- Point p_3 is both sparse and distant from the center, likely to be an outlier and thus should be assigned a low weight.

3.1.1. Constructing the weight function

In the following, $F(x)$ calculates the weight of an instance x in a class. For this purpose, first, at a fixed radius $\epsilon > 0$, the number of points, N_x in the neighborhood of a x is determined by its "mass". Thus, $N_{\bar{x}}$ is the mass of $\bar{x}$, the center of the class. Empirically p is set to $\frac{1}{2}$ [22]. In $F(x)$, we call the parameter G the value of gravity between x and $\bar{x}$.

$$F(x) = G \times \frac{N_{\bar{x}} \times\ N_x}{(\delta + r)^p}. \tag{14}$$

As mentioned in relation Eq. (13), r is the Euclidean distance between x and $\bar{x}$ and that value is zero for $\bar{x}$, If $x = \bar{x}$, the $\delta > 0$ is used to prevent the denominator from becoming zero. To overcome class imbalance, we first assume that the smaller class is as important as the larger class. From relation Eq. (8) it follows that $IR \times N(A) = N(B)$. Using this fact in the weight calculation, It can partially overcome the class imbalance similar to the virtual data generation in A. For further scaling, Here unlike Newton's universal gravitation, the value of G is not constant, its value is different in each class and, on the other hand, it depends on a hyperparameter $g > 0$. We define it as follows:

$$G = \begin{cases} g \times IR, & \text{if } x \in A \\ g, & \text{if } x \in B. \end{cases} \tag{15}$$

By default, $g = 1$ is considered.

[1] Isaac Newton (1643-1727): Discoverer of the universal gravitation.

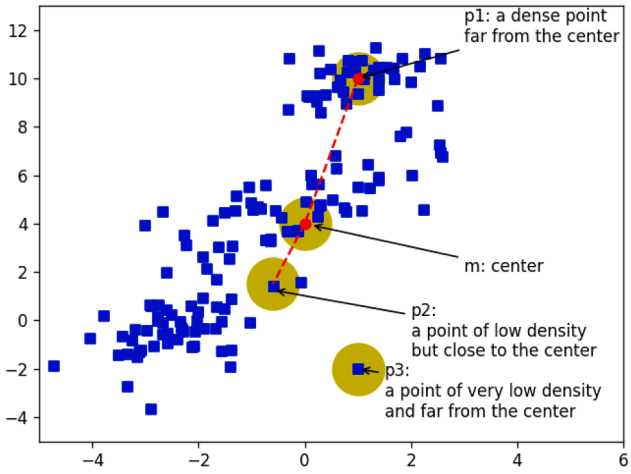

Fig. 1. Intuitive model inspired by gravitational force for weighting data points.

3.1.2. Properties of the weight function

1) A small value of r means that the point is very close to the center and should have a higher weight, and theoretically, Eq. (14) would meet this expectation and produce a larger weight. Similarly, points further from the center correspond to a large value of r and produce larger weights. If we call the distance of the farthest point from the center $r_{\max}$ then $0 \le r \le r_{\max}$.
2) If a point has a lot of accumulation around it and is close to the center of the class, it has a very high weight.
3) If a point has very little accumulation around it and is far from the center of the class, it has a very low weight.
4) If a point has a lot of accumulation and is far from the center, it has more gravity compared to the quiet points in this distance
5) Points with low density but close to the center have average weight, which is also logically acceptable because the points close to the center, even though they are lonely points, are probably members of the class.
6) Noisy data are data that have few neighboring points and are far from the center.
7) The highest and lowest value of weight may occur for points of both classes, but in ideal conditions, the highest value occurs for samples very close to the center, This happens when $r \approx 0$ and $N_x \approx N_{\bar{x}}$, therefore $F(x) \approx G \times N_x^2$. The lowest value also occurs for the farthest points from the center with a distance of $r_{\max}$, which are isolated points in a specified radius. By setting $g = 1$ if $N_x = 1$, in ideal conditions it is equal to $F(x) \approx G \times N_{\bar{x}}$. So we have:

$$G \times N_{\bar{x}} \le F(x) \le G \times N_{\bar{x}}^2.$$

If the dataset is balanced according to Eq. (8), IR is approximately equal to 1 and since $g = 1$ is assumed from Eq. (15) the value of G for the small class is $G = g \times IR \approx 1$ and for the large class is $G = g = 1$. So we can write $N_{\bar{x}} \le F(x) \le N_{\bar{x}}^2$. It is emphasized that this relationship can be valid for ideal conditions.

3.1.3. Algorithm for calculating the weight of samples

Suppose C is the data of positive or negative class (subset of A or B in the defination Eq. (1)) and G is given by Eq. (15), we calculate the gravity weights of the points using the Algorithm 1.

3.2. Linear GLSTSVM-CIL

Assume that D is the dataset Eq. (1) defined in Section 2 containing m samples with n features, $A \in \mathbb{R}^{m_1 \times n}$ positive class matrix consisting of m_1 training samples and $B \in \mathbb{R}^{m_2 \times n}$ is the negative class matrix consisting of m_2 training samples, the first optimization problem is defined as follows:

$$\begin{aligned} &\min \frac{c_1}{2}(||w_1||^2 + b_1^2) + \frac{c_2}{2}(S_1\eta_1)^T(S_1\eta_1) + \frac{c_3}{2}(S_2\xi_1)^T(S_2\xi_1) \\ &S.t. \quad Aw_1 + b_1 e = \eta_1, \\ &-(Bw_1 + b_1 e) + \xi_1 = e. \end{aligned} \tag{16}$$

and second problem as follows:

$$\begin{aligned} &\min \frac{c_1}{2}(||w_2||^2 + b_2^2) + \frac{c_2}{2}(S_2\eta_2)^T(S_2\eta_2) \\ &+ \frac{c_3}{2}(S_1\xi_2)^T(S_1\xi_2) + \frac{1}{2}\gamma^2 \end{aligned}$$

Algorithm 1 : Compute weights of the samples in matrix $C_{n\times n}$.

INPUTE: Matrix $C_{n\times n}$, parameter G
OUTPUT: Diagonal matrix S

Initalize $N_c = 1$
Initalize S as identity matrix $n \times n$
Compute center of C as $\bar{x}$
Compute D as vector of distance from $\bar{x}$
For $i \leq n$
  $x_i := C[i]$
  if $D_i < \epsilon$ then $N_c := N_c + 1$
  For j:= i to n
    $x_j := C[j]$
    If $\|xi - xj\| < \epsilon$ then
      $S(i,i) := S(i,i) + 1$
      $S(j,j) := S(j,j) + 1$
  $d := (\delta + D(i))^{\frac{1}{2}}$
  $S(i,i) := \frac{S(i,i)}{d}$
$S := G \times N_c \times S$
Return S as weight matrix of the samples in C.

$$
\begin{aligned}
&Bw_2 + b_2 e = \eta_2,\\
&(Aw_2 + b_2 e) + \xi_2 = e,\\
&w_1^T w_2 + b_1 b_2 = \gamma.
\end{aligned} \tag{17}
$$

where $w_1, w_2 \in R^n$ and $b_1, b_2 \in R$ are bias values, also w_1 and b_1 are solved in Eq. (16). In Section 3.1, it was said that each point of the small class must have an equivalent value to the points of the larger class, To implement this concept we consider the parameters that play the same role in both problems to be equal. In other words, we penalize both hyperplanes equally (this does not include the weight function because the weight function assigns a special weight to each point separately). Therefore, in this problem, only three parameters c_1, c_2 and c_3 are used. The first term maximizes the hypothesized margins of SVM. In both objective functions, the parameter c_1 weights both problems equally. According to the first constraint, the second term minimizes the sum of squared distances of the points from each hyperplane. $\eta_1 \in R^{m_1}$ and $\eta_2 \in R^{m_2}$ are the error vectors of the points, which are penalized by the matrices S_1 and S_2. The second constraint in each case, in cooperation with the third term, keeps the points of each class at a distance of one unit from the hyperplane of the other class. Any difference in this distance, with the vectors $\xi_1 \in \mathbb{R}^{m_2}$ and $\xi_2 \in R^{m_1}$ are given. The diagonal matrix $S_1 \in R^{m_1 \times m_1}$ is the values of Gravity function of the smaller class so that $F(x_i) = S_1(i,i)$ and $x_i \in A$. Similarly, the diagonal matrix $S_2 \in R^{m_2 \times m_2}$ also shows the values of Gravity function of the larger class. The second and third terms in both objective functions are penaltized by c_2 and c_3, respectively. The term $\frac{1}{2}\gamma^2$ in the objective function of the second problem optimizes the angle of the hyperplanes. This term, together with the first term, increases the structural risk control of the system. $c_1 > 0$ is structural risk parameter and $c_2, c_3 > 0$ are error parameters. The vectors e, in each position are vectors of **1** and dimensions proportional to that position. To solve problem Eq. (16), we first form its Lagrangian function:

$$
\begin{aligned}
L_1 = &\frac{c_1}{2}(||w_1||^2 + b_1^2) + \frac{c_2}{2}(S_1\eta_1)^T(S_1\eta_1)\\
&+ \frac{c_3}{2}(S_2\xi_1)^T(S_2\xi_1) + \alpha_1^T(\eta_1 - Aw_1 - b_1 e)\\
&+ \alpha_2^T(\xi_1 - Bw_1 - b_1 e - e),
\end{aligned} \tag{18}
$$

where α_1 and α_2 are vectors of Lagrangian coefficients. By deriving from L_1 and setting it to zero, the following relationships are obtained:

$$\frac{\partial L_1}{\partial w_1} = c_1 w_1 - A^T\alpha_1 - B^T\alpha_2 = 0, \tag{19}$$

$$\frac{\partial L_1}{\partial b_1} = c_1 b_1 - e^T\alpha_1 - e^T\alpha_2 = 0, \tag{20}$$

$$\Longrightarrow \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \frac{1}{c_1}\begin{bmatrix} A^T & B^T \\ e^T & e^T \end{bmatrix}\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}. \tag{21}$$

$$\frac{\partial L_1}{\partial \eta_1} = c_2 S_1^2\eta_1 + \alpha_1 = 0 \Longrightarrow \eta_1 = \frac{-1}{c_2}S_1^{-2}\alpha_1, \tag{22}$$

$$\frac{\partial L_1}{\partial \xi_1} = c_3 S_2^2\xi_1 + \alpha_2 = 0 \Longrightarrow \xi_1 = \frac{-1}{c_3}S_2^{-2}\alpha_2, \tag{23}$$

$$\frac{\partial L_1}{\partial \alpha_1} = \eta_1 - Aw_1 - eb_1 = 0$$

$$\Longrightarrow \frac{1}{c_2} S_1^{-2}\alpha_1 + \begin{bmatrix} A & e \end{bmatrix} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = 0. \tag{24}$$

$$\frac{\partial L_1}{\partial \alpha_2} = \xi_1 - Bw_1 - eb_1 - e = 0$$

$$\Longrightarrow \frac{1}{c_3} S_2^{-2}\alpha_2 + \begin{bmatrix} B & e \end{bmatrix} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -e. \tag{25}$$

By combining the above relations and multiplying both sides of the equations in c_1 and rewriting in matrix form, we have:

$$\left(\begin{bmatrix} AA^T + \frac{c_1}{c_2} S_1^{-2} & AB^T \\ BA^T & BB^T + \frac{c_1}{c_3} S_2^{-2} \end{bmatrix} + E \right) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = -c_1 \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}, \tag{26}$$

where $\mathbf{0}$ is a zero vector, dimension of α_1 and $\mathbf{1}$ is a vector of ones that is dimension of α_2. After solving the Eq. (26), the parameters of the model are obtained through Eq. (21).

Now, to solve the second problem, similar to the previous one, we write its Lagrangian function. According to Eq. (17) we have:

$$\begin{aligned} L_2 = {} & \frac{c_1}{2}(\|w_2\|^2 + b_2^2) + \frac{c_2}{2}(S_2\eta_2)^T(S_2\eta_2) + \frac{c_3}{2}(S_1\xi_2)^T(S_1\xi_2) \\ & + \frac{1}{2}\gamma^2 + \beta_1^T\left(\eta_2 - Bw_2 - eb_2\right) + \beta_2^T\left(e - Aw_2 - eb_2 - \xi_2\right) \\ & + \beta_3(\gamma - w_1^T w_2 - b_1 b_2), \end{aligned} \tag{27}$$

where β_1, β_2 are vectors and β_3 are scalar, all Lagrangian coefficients. By deriving and setting the derivatives to zero, we have:

$$\frac{\partial L_2}{\partial w_2} = c_1 w_2 - B^T\beta_1 - A^T\beta_2 - \beta_3 w_1 = 0, \tag{28}$$

$$\frac{\partial L_2}{\partial b_2} = c_1 b_2 - e^T\beta_1 - e^T\beta_2 - \beta_3 b_1 = 0 \tag{29}$$

$$\Longrightarrow \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \frac{1}{c_1} \begin{bmatrix} B^T & A^T & w_1 \\ e^T & e^T & b_1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}. \tag{30}$$

$$\frac{\partial L_2}{\partial \eta_2} = c_2 S_2^2\eta_2 + \beta_1 = 0 \Longrightarrow \eta_2 = \frac{-1}{c_2} S_2^{-2}\beta_1, \tag{31}$$

$$\frac{\partial L_2}{\partial \xi_2} = c_3 S_1^2\xi_2 - \beta_2 = 0 \implies \xi_2 = \frac{1}{c_3} S_1^{-2}\beta_2, \tag{32}$$

$$\frac{\partial L_2}{\partial \gamma} = \gamma + \beta_3 = 0 \implies \gamma = -\beta_3, \tag{33}$$

$$\frac{\partial L_2}{\partial \beta_1} = \eta_2 - Bw_2 - b_2 e = 0$$

$$\Longrightarrow \frac{-1}{c_2} S_2^{-2}\beta_1 + \begin{bmatrix} B & e \end{bmatrix} \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = 0. \tag{34}$$

$$\frac{\partial L_2}{\partial \beta_2} = e - \xi_2 - Aw_2 - b_2 e = 0$$

$$\Longrightarrow \frac{-1}{c_3} S_1^{-2}\beta_2 + \begin{bmatrix} A & e \end{bmatrix} \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = -e. \tag{35}$$

$$\frac{\partial L_2}{\partial \beta_3} = \gamma - w_1^T w_2 - b_1 b_2 = 0$$

$$\Longrightarrow \gamma - \begin{bmatrix} w_1 & b_1 \end{bmatrix} \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = 0. \tag{36}$$

Using Eq. (30) we have:

$$\frac{1}{c_2} S_2^{-2}\beta_1 + \frac{1}{c_1} \begin{bmatrix} BB^T + E_1 & BA^T + E_2 & Bw_1 + b_1 e \end{bmatrix} \beta = 0, \tag{37}$$

$$\frac{1}{c_3} S_1^{-2}\beta_2 + \frac{1}{c_1} \begin{bmatrix} AB^T + E_2^T & AA^T + E_3 & Aw_1 + b_1 e \end{bmatrix} \beta = e, \tag{38}$$

$$\beta_3 + \frac{1}{c_1} \begin{bmatrix} Bw_1 + b_1 e^T & Aw_1 + b_1 e^T & w_1^T w_1 + b_1^2 \end{bmatrix} \beta = 0, \tag{39}$$

where $\beta = [\beta_1 \quad \beta_2 \quad \beta_3]^T$. By combining the above three relations and multiplying both sides of the equations in $-c_1$ and rewriting in matrix form, we have:

$$Q\beta = c_1 \begin{bmatrix} \mathbf{0} & e & 0 \end{bmatrix}^T. \tag{40}$$

where

$$Q = \begin{bmatrix} BB^T + \frac{c_1}{c_2} S_2^{-2} + E_1 & BA^T + E_2 & Bw_1 + b_1 e \\ AB^T + E_2^T & AA^T + \frac{c_1}{c_3} S_1^{-2} + E_3 & Aw_1 + b_1 e \\ w_1 B^T + b_1 e^T & w_1 A^T + b_1 e^T & w_1^T w_1 + b_1^2 \end{bmatrix}$$

and $\mathbf{0}$ is zero vector, e are vectors of Ones and E_i are matrices of Ones, all is used with suitable dimensions. by solving the system Eq. (40) the Lagrangian coefficients are obtained, which are put in the relation Eq. (30) can also obtain the second hyperplane. Having w_1, w_2, b_1, b_2 the class of the new instance can be predicted as follows:

Decision function: for each input x, the decision function is implemented as:

$$f_i(x) = w_i^T x + b_i. \tag{41}$$

Thus class of each x is determined by:

$$class(x) = \begin{cases} Positive & |f_1(x)| < |f_2(x)| \\ Negative & Otherwise. \end{cases} \tag{42}$$

The angle between hyperplanes: In linear model, the angle between two hyperplanes can be calculated. Let $\vec{N}_1$ and $\vec{N}_2$ be the norm vectors of hyperplanes and θ be the angle between two hyperplanes. This angle can be obtained by calculating the angle between norm vectors. Here vectors are $\vec{w}_1$ and $\vec{w}_2$, so:

$$\cos(\theta) = \frac{\vec{w}_1 . \vec{w}_2}{||\vec{w}_1|| ||\vec{w}_2||}. \tag{43}$$

The sum of squared distances: Also, by using η_1 and η_2, we can calculate the sum of squared distances of the points from each hyperplane. Using the first constraint of Eqs. (16) and (17) the sum of squared distances can be defined as follows:

$$SSD = \sqrt{\eta_1^T \eta_1 + \eta_2^T \eta_2}. \tag{44}$$

This value shows the sum of squared distances of all points obtained from both hyperplanes, first term is sum of squared distances of point of first hiperplane and second term is about to second hyperplane.

3.3. Nonlinear GLSTSVM-CIL

When the data are linearly inseparable, the classifiers based on SVM and later versions use kernel functions, the first nonlinear problem of GLSTSVM-CIL is written as follows:

$$\begin{aligned} &min \ \frac{c_1}{2}(\|w_1\|^2 + b_1^2) + \frac{c_2}{2}(S_1\eta_1)^T(S_1\eta_1) + \frac{c_3}{2}(S_2\xi_1)^T(S_1\xi_1) \\ &S.t. \ \ \phi(A)w_1 + b_1 e = \eta_1, \\ &\quad -(\phi(B)w_1 + b_1 e) + \xi_1 = e, \end{aligned} \tag{45}$$

and second nonlinear problem is written as:

$$\begin{aligned} &\min \frac{c_1}{2}(\|w_2\|^2 + b_2^2) + \frac{c_2}{2}(S_2\eta_2)^T(S_2\eta_2) + \frac{c_3}{2}(S_1\xi_2)^T(S_1\xi_2) + \frac{1}{2}\gamma^2 \\ &S.t \ \ \phi(B)w_2 + eb_2 = \eta_2, \\ &\quad \phi(A)w_2 + eb_2 + \xi_1 = e, \\ &\quad w_1^T w_2 + b_1 b_2 = \gamma, \end{aligned} \tag{46}$$

which $\phi(.)$ was defined in Eq. (10).

The two optimization problems-LS-FLSTSVM-CIL and GLSTSVM-CIL-exhibit important mathematical differences that reflect their underlying design philosophies and optimization strategies. the LS-FLSTSVM-CIL model introduces three main components in its objective function: (i) a regularization term ($\|w_i\|^2 + b_i^2$) to control model complexity and enforce structural risk minimization, (ii) a squared fitting error $\|\eta_i\|^2$ to reduce the distance between intra-class samples and the corresponding hyperplane, and (iii) a fuzzy-weighted penalty term $(S_j\xi_j)^T(S_j\xi_j)$ that S_1 is identity matrix and S_2 is diagonal matrix containing the fuzzy membership values at the diagonal places [10,14].

In contrast, the GLSTSVM-CIL formulation applies weight matrices S_1 and S_2 to both intra-class fitting errors η_i and inter-class slack variables ξ_i, these weights are not fuzzy in nature and is computed using gravity function Eq. (14). In addition, GLSTSVM-CIL introduces a term $\frac{1}{2}\gamma^2$ in the objective function and a constraint $w_1^T w_2 + b_1 b_2 = \gamma$ in the second optimization problem, which enforces a geometric relationship between the two hyperplanes. This constraint aims to increase the overall separation and complementarity of the hyperplanes, potentially improving the generalization performance.

To solve GLSTSVM-CIL by forming the Lagrangian function of Eq. (45) we have:

$$L_1 = \frac{c_1}{2}(||w_1||^2 + b_1^2) + \frac{c_2}{2}(S_1\eta_1)^T(S_1\eta_1) + \frac{c_3}{2}(S_2\xi_1)^T(S_2\xi_1) + \alpha_1^T(\eta_1 - \phi(A)w_1 - b_1 e) + \alpha_2^T(\xi_1 - \phi(B)w_1 - b_1 e - e). \tag{47}$$

By calculating the $\frac{\partial L_1}{\partial w_1} = 0$ and $\frac{\partial L_1}{\partial b_1} = 0$ we obtain:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \frac{1}{c_1}\begin{bmatrix} \phi^T(A) & \phi^T(B) \\ e^T & e^T \end{bmatrix}\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}. \tag{48}$$

By calculating the partial derivatives of η_1, ξ_1, α_1 and α_2 and setting them equal to zero, by following a process similar to the linear case, the following equation can be reached:

$$\left(\begin{bmatrix} K(A,A^T) + \frac{c_1}{c_2}S_1^{-2} & K(A,B^T) \\ K(B,A^T) & K(B,B^T) + \frac{c_1}{c_3}S_2^{-2} \end{bmatrix} + E\right)\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = -c_1\begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}. \tag{49}$$

And the Lagrangian function of the second nonlinear problem is written as follows:

$$L_2 = \frac{c_1}{2}(\|w_2\|^2 + b_2^2) + \frac{c_2}{2}(S_2\eta_2)^T(S_2\eta_2) + \frac{c_3}{2}(S_1\xi_2)^T(S_1\xi_2) + \frac{1}{2}\gamma^2 + \beta_1^T\left(\eta_2 - \phi(B)w_2^T - eb_2\right) + \beta_2^T\left(e - \phi(A)w_2^T - eb_2 - \xi_2\right) + \beta_3(\gamma - w_1 w_2^T - b_1 b_2). \tag{50}$$

By calculating the $\frac{\partial L_2}{\partial w_2} = 0$ and $\frac{\partial L_2}{\partial b_2} = 0$ we obtain:

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \frac{1}{c_1}\begin{bmatrix} \phi^T(B) & \phi^T(A) & w_1 \\ e^T & e^T & b_1 \end{bmatrix}\begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}. \tag{51}$$

By calculating the partial derivatives of $\eta_2, \xi_2, \gamma, \beta_1, \beta_2$ and β_3 and setting them equal to zero, the following equation can be reached:

$$Q\beta = c_1\begin{bmatrix} \mathbf{0} & e & 0 \end{bmatrix}^T. \tag{52}$$

where:

$$Q = \begin{bmatrix} K(B,B^T) + \frac{c_1}{c_2}S_2^{-2} + E_1 & K(B,A^T) + E_2 & \phi(B)w_1 + b_1 e \\ K(A,B^T) + E_2^T & K(A,A^T) + \frac{c_1}{c_3}S_1^{-2} + E_3 & \phi(A)w_1 + b_1 e \\ w_1^T\phi^T(B) + b_1 e^T & w_1^T\phi^T(A) + b_1 e^T & w_1^T w_1 + b_1^2 \end{bmatrix}$$

By solving the problem Eq. (52), β_1, β_2 and β_3 are obtained as before, which are put in the relation Eq. (51) the second hyperplane can be obtained. Considering that the function $\phi(.)$ is unknown in the last column, we first note that the vector $[w_1 \quad b_1]^T$ is available through Eq. (48), so we use the following trick to get column 3. First it should be noted that

$$\phi(B)w_1 + eb_1 = \begin{bmatrix} \phi(B) & e \end{bmatrix}\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}. \tag{53}$$

Now we take help from Eq. (48) and by multiplying Eq. (48) by $\begin{bmatrix} \phi(B) & e \end{bmatrix}$ we reach the following useful relation:

$$\begin{bmatrix} \phi(B) & e \end{bmatrix}\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \frac{1}{c_1}\begin{bmatrix} K(B,A^T) + E & K(B,B^T) + E \end{bmatrix}\alpha. \tag{54}$$

It can be repeated for other elements of this column:

$$\begin{bmatrix} \phi(A) & e \end{bmatrix}\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \frac{1}{c_1}\begin{bmatrix} K(A,A^T) + E & K(A,B^T) + E \end{bmatrix}\alpha. \tag{55}$$

$$\begin{bmatrix} w_1^T & b_1 \end{bmatrix}\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \frac{1}{c_1^2}\alpha^T\left(\begin{bmatrix} K(A,A^T) & K(A,B^T) \\ K(B,A^T) & K(B,B^T) \end{bmatrix} + E\right)\alpha. \tag{56}$$

where $\alpha = [\alpha_1^T \quad \alpha_2^T]^T$. using Eqs. (54), (55) and (56), the third column of the Q is obtained, for the third row, it is enough to put the transpose of the third column.

Decision function: decision function of first hyperplane for nonlinear case is defined as:

$$f_1(x) = w_1^T \phi(x) + b_1 \Longrightarrow$$
$$f_1(x) = \frac{1}{c_1}\left[K(x, A^T) + e\ ,\quad K(x, B^T) + e\right]\alpha. \tag{57}$$

And for second hyperplane we have:

$$f_2(x) = w_2^T \phi(x) + b_2 \Longrightarrow$$
$$f_2(x) = \frac{1}{c_1}\left[K(x, B^T) + e,\quad K(x, A^T) + e,\quad w_1^T \phi(x) + b_1\right]\beta. \tag{58}$$

The last element of function f_2 is equal to f_1, so we have:

$$f_2(x) = \frac{1}{c_1}\left[K(x, B^T) + e\quad K(x, A^T) + e\quad f_1(x)\right]\beta. \tag{59}$$

From Eqs. (57) and (59), label of new instance x in detected by:

$$class(x) = \begin{cases} Positive & |f_1(x)| < |f_2(x)|, \\ Negative & Otherwise. \end{cases} \tag{60}$$

4. Solution method

The final matrices in both linear and non-linear problems are symmetric matrices. As discussed in the methodology section, the stable and efficient convergence of the Conjugate Gradient(CG) method requires the matrix Q to be positive definite. A detailed proof of the positive definiteness of matrix Q is provided in Appendix A that supports fundamental for guaranteeing the convergence and numerical stability of the Conjugate Gradient method used in our framework. Thus the solution to Eqs. (26), (40), (49) and (52) determines the global optimal solution [23] and instead of solving a QPP, we just need to solve a linear system Ax = b. In order to Proposed Method works with large-scale data, we use a numerical method. Among the many methods that exist in this field, conjugate gradient methods are more suitable. This method is used to solve Ax = b systems where A is symmetric and positive definite [24,25] and are faster than steepest descent methods [26]. The ability to solve with large-scale matrices, low memory usage and suitable speed are among the reasons that motivated us to choose this method, an implementation of this algorithm is given in Algorithm 2.

Algorithm 2 :Conjugate gradient method to solve $Ax = b$.

INPUTE: Matrix A, vector b, max_iter and $tolorance > 0$
OUTPUT: Solution of x

1: Initialize $x_0 = \vec{0}, r_0 := b - Ax_0, p_0 := r_0$
2: For $i \leq \max\_iter$
3: Compute the step size, $\alpha = \frac{r_k^T r_k}{p_k^T Q p_k}$
4: Update the solution, $x_{k+1} := x_k + \alpha p_k$
5: If $\|x_{k+1} - x_k\| < tolerance$ terminate by x_{k+1} as optimal solution
6: Update the residual, $r_{k+1} := r_k - \alpha Q p_k$
7: Update the search direction $p_{k+1} := r_{k+1} + \frac{\|r_{k+1}\|}{\|r_k\|} p_k$
8: Return x_{k+1} as optimal solution.

4.1. Computational complexity

In Algorithm 1, for computing the gravity function matrix, a loop with m_1 iterations is used for the smaller class, which has a computational complexity of $O(m_1)$. Within this loop, the neighborhood of points with the class center is controlled. The inner loop runs $\frac{m_1 \times (m_1 - 1)}{2}$ times to count the number of neighborhoods for each point. This part has a complexity of $O(m_1^2)$. The code then calculates the weight of each sample, which has a complexity of $O(m_1)$. In total, considering the mentioned parts, the computational complexity of the gravity function is $O(m_1^2)$. Similar results are obtained when considering the gravity matrix computations for the negative class. Therefore, the overall complexity is $O(m_1^2) + O(m_2^2)$.

Additionally, two systems in the form of $Qx = c$ (represented as $Ax = b$ in general CG literature) are solved using the Conjugate Gradient (CG) algorithm. Generally, this algorithm has a complexity of $O(m_0\sqrt{\kappa})$, where m_0 is the number of non-zero elements in the matrix A (here, Q) and κ is its condition number, given by $\kappa = \frac{|\text{largest eigenvalue}|}{|\text{smallest eigenvalue}|}$ [27,28].

Considering the Eqs. (1) and (26), (40), (49), and (52), the matrix A corresponds to the matrix Q in these equations. In the first system, the matrix Q is of size $m \times m$, and in the second system, it is of size $(m+1) \times (m+1)$. Therefore, if we consider the condition number of the first problem as κ_1 and the number of non-zero elements as m_0 (which is $O(m^2)$ for a dense matrix like Q), the overall complexity of the first problem's solution by CG would be $O(m_0\sqrt{\kappa_1})$. Combined with the gravity function calculation, its total complexity becomes $O(m_0\sqrt{\kappa_1}) + O(m_1^2)$.

From another perspective, focusing on the iterative process outlined in Algorithm 2, the CG algorithm takes the matrix Q (with dimensions $m \times m$ or $(m+1) \times (m+1)$) as input, along with a maximum number of iterations, **max_iter**. In each iteration, the dominant operation is a matrix-vector multiplication. Given the size of matrix Q, these computations in the first system are of order $O(m^2)$. Scalar-vector computations and conditional statements are of order $O(m)$. Therefore, if the loop iterates **max_iter** times, the complexity of CG is $O(\text{max\_iter} \times m^2)$. This must be added to the complexity of the weight function. Thus, the complexity of the first system is $O(\text{max\_iter} \times m^2) + O(m_1^2)$. By repeating a similar reasoning for the second problem, the complexity $O(\text{max\_iter} \times (m+1)^2) + O(m_2^2)$ is obtained.

4.2. CG convergence for equation (52)

The effectiveness and practical convergence of the Conjugate Gradient (CG) method for solving Eq. (52) are profoundly influenced by the properties of the matrix Q. For CG to guarantee convergence and numerical stability, Q must be symmetric and positive definite (SPD) [27,28]. As established in Appendix A: Proof of Positive Definiteness for Matrix Q, our matrix Q indeed satisfies these crucial requirements. The specific structure of Q significantly impacts CG's convergence rate:

- **Guaranteed Convergence and Uniqueness:** Since Q is proven to be positive definite, the quadratic objective function associated with $Qx = b$ ($f(x) = \frac{1}{2}x^TQx - x^Tb$) is strictly convex. This strict convexity ensures a unique global minimum for the function, which corresponds precisely to the unique solution of $Qx = b$ [29]. CG's fundamental mechanism minimizes this objective function, ensuring convergence to this unique solution.
- **Enhanced Rate of Convergence from Q's Structure:** While CG theoretically converges in a finite number of steps (N iterations, where N is Q's dimension), the practical speed of convergence depends heavily on the condition number ($\kappa(Q)$) of the matrix. A lower $\kappa(Q)$ leads to faster convergence. Our matrix Q benefits from specific terms that accelerate CG's performance:
  - **Kernel Matrix Properties:** The $K(X, Y)$ blocks are inherently positive semidefinite [30], forming the structural basis of the system.
  - **Built-in Regularization from S_i:** The terms $\frac{c_1}{c_2}S_2^{-2}$ and $\frac{c_1}{c_3}S_1^{-2}$ are critical. As detailed in Appendix A, these are strictly positive diagonal matrices. Their direct addition to the diagonal of the kernel blocks acts as a powerful Tikhonov regularization [31]. This regularization is vital because it:
    - Improves the Condition Number: By adding positive values to the diagonal, these terms increase Q's smallest eigenvalue, thereby reducing its condition number ($\kappa(Q)$). A lower $\kappa(Q)$ directly translates to a significantly faster convergence rate for CG. This is especially beneficial if the original kernel matrices or the combined system would otherwise be ill-conditioned.
    - Enhances Numerical Stability: Regularization makes the system $Qx = c$ more robust to numerical errors during the iterative process.
  - **Positive Contribution from w_1 and b_1:** The final diagonal term, $w_1^Tw_1 + b_1^2$, is strictly positive, further contributing to Q's overall positive definiteness across all dimensions.

In essence, the careful design of matrix Q, particularly its inherent SPD property strengthened by explicit regularization terms from S_i, ensures that the Conjugate Gradient method for solving Eq. (52) will converge not only reliably to a unique solution but also with an improved and robust rate, which is paramount for practical computational efficiency.

5. Experimental results

In this section, the performance of Proposed Method has been evaluated in comparison with FTSVM [6], LSTSVM [4], LS-ATWSVM [5], SVM-Based SA-Method [8] and LSFLSTSVM-CIL [10] . First, using two-dimensional synthetic data, the position of hyperplanes and the performance of the method against changes in the imbalance ratio are evaluated. KEEL repositoy was used to evaluate the performance on imbalance and noisy datasets [32]. The data available in UCI [33] and Kaggle[2] repositories were used to evaluate medical applications. Text data from UCI, Kaggle

[2] https://www.kaggle.com

Table 1
Parameter settings.

Method	Parameters	Options
Proposed Method	c_1, c_2, c_3, μ, g	$c_i, \mu \in \{10^j, j = -5, \dots, 5\}$, $g \in \{\frac{j}{10}, j = 1, \dots, 9\} \cup \{1, \dots, 10\}$
LSFLSTSVM-CIL	$c_0, c_1 = c_2, c_3 = c_4, \mu$	$c_0 \in \{0.5, 1, 1.5, 2, 2.5\}, c_1, c_2, c_3, c_4, \mu \in \{10^j, j = -5, \dots, 5\}$
LS-ATWSVM	$c_1, c_2, c_3, c_4 = 1 - c_2, c_5 = c_1, \mu$	$c_1, c_3 \in (0, 1], c_2 \in (0, 1), \mu \in \{10^j, j = -5, \dots, 5\}$
FTSVM	$c_1 = c_2, \mu$	$c_1, c_2, \mu \in \{10^j, j = -5, \dots, 5\}$
LSFTSVM	$c_1 = c_2, \mu$	$c_1, c_2, \mu \in \{10^j, j = -5, \dots, 5\}$

Table 2
System specifications.

Processor	11th Gen Intel(R) Core(TM) i5-1135G7
	@ 2.40GHz 2.42 GHz
Installed RAM	8.00 GB (7.70 GB usable)
System type	64-bit operating system, x64-based processor
Edition	Windows 11 Pro
Version	23h2
OS build	22631.3447

Table 3
System specifications for NDC datasets.

Processor	Intel(R) Core(TM) i7-4960X CPU
	@ 3.60GHz 3.60 GHz @ 2.40GHz 2.42 GHz
Installed RAM	64.00 GB
System type	64-bit operating system, x64-based processor
Edition	Windows 10 Pro

and TDT2 dataset [34] have been used to evaluate models in text data classification. NDC data [35] was used to evaluate the performance against large-scale data. Gaussian kernel $K(X, Y) = e^{\mu \|X - Y\|^2}$, linear kernel $K(X, Y) = X^T Y$ were used to transform the data into kernel space. Also, sigmoid kernel $K(X, Y) = \tanh\left(\gamma X^T Y\right)$ was used in text data. In all evaluations, the k-folds method with k = 5 was used to select the best parameters. For a fair comparison, all equivalent parameters were selected according to [5,10]. The tolorance value for solving LSFLSTSVM-CIL and Proposed Method is fixed to 10^{-5} and the number of repetitions were fixed to 10^2. Table 1 shows the parameter setting pattern and their range completely. The parameter $g \in \{1, \dots, 10\}$ in the proposed model scales the weight values j-times larger or $\frac{j}{10}$-times smaller when $g \in \{\frac{j}{10}, j = 1, \dots, 9\}$.

Also In each case, the data is randomly divided into training and testing parts with a ratio of 70-30, and the parameters are obtained through Grid-Search and cross-validation methods [36], and to run the experiments, Python 3.12.2 in the Visual Studio code 1.92.0 environment is used. All evaluations except NDC data was done on the personal laptop with the specifications in Table 2. Also, to evaluate NDC data, a computer with specifications in Table 3 was used in the computer laboratory of the Faculty of Mathematical Sciences of Ferdowsi University of Mashhad. Scipy 1.14.0 library has been used to solve Proposed Method with the conjugate gradient algorithm.

The following evaluation criteria is used to evaluate the performance of the methods:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{61}$$

$$Recall = \frac{TP}{TP + FN}, \tag{62}$$

$$Precision = \frac{TP}{TP + FP}, \tag{63}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precisoin + Recall}, \tag{64}$$

$$G - mean = \sqrt{Precision \times Recall}. \tag{65}$$

TP, TN, FP, FN respectively are: True Positive, True Negative, False Positive, False Negative. In all evaluations, the smaller class was considered equivalent to the positive class.

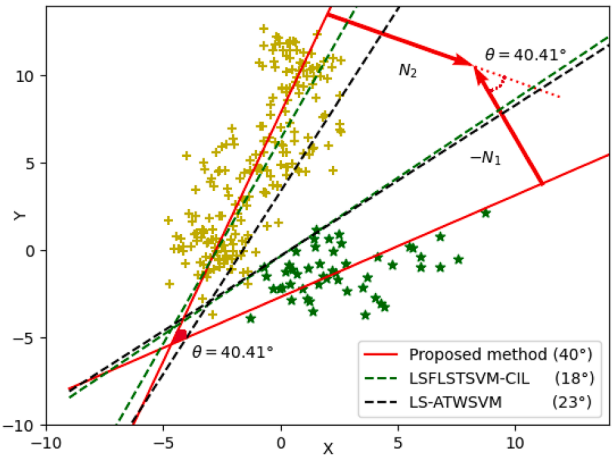

Fig. 2. Position of hyperplanes in LS-ATWSVM, LSFLSTSVM-CIL, GLSTSVM-CIL.

Table 4
Numerical results related to the Fig. 2.

Methods	SSD	Angle	Accuracy	F1
Proposed Method	8.55	40.41	100	100
LSFLSTSVM-CIL	19.23	18.82	98.80	98.04
LS-ATWSVM	17.87	23.68	97.59	96.15

5.1. Experimental results on synthetic data

5.1.1. Hyperplanes

One of the important points in the methods developed on the basis of TSVM is how the hyperplanes are placed in relation to the instances of each class. The hyperplanes obtained from Proposed Method, LS-ATWSVM and LSFLSTSVM-CIL in the Fig. 2 and the numerical results related to this figure, in the Table 4 is shown. In this study, a synthetic dataset with normal distribution is used, structural risk parameters are equal to 1 in all three methods and other parameters are obtained using Grid-Search.

In the Fig. 2, the hyperplanes of Proposed Method and LSFLSTSVM-CIL in the larger class are close to each other, and LS-ATWSVM has a significant deviation from these two is placed. The reason for this observation can be that the first two methods give weight to large class points, while LS-ATWSVM does not have a weight function. On the other hand, LSFLSTSVM-CIL does not generate weights for the smaller class to avoid slowing down the algorithm speed, so its hyperplane is expected to be similar to LS-ATWSVM on the smaller class side, as shown in Fig. 2 It can be seen that their hyperplane is very close to each other in this area.

Proposed Method emphasizes the importance of smaller class points and in its optimization problem, weight is also assigned to these points. On the other hand, like LS-ATWSVM, by using a term to optimize the angle, it also improves structural risk control. It seems that this action has made its hyperplane to have a better position on the side of the smaller class and is closer to the class points than the other two methods. According to the Table 4, it can be seen that the sum of squared distances(SSD) of the points in this method is less than the others, which confirms that the hyperplane is close to the class points.

5.1.2. Noisy data with imbalance rate

In order to further investigate the effect of class imbalance ratio on the performance of the methods, the three models were tested on binary classification synthetic data. For this purpose, first, 20 *normally distributed* datasets were created using the *sklearn* library tools, in which the imbalance ratio was changed from 1 to 20. Each dataset contained 350 samples, and the number of minority class samples decreased with increasing imbalance ratio. Then, to simulate noisy conditions, Gaussian noise with mean $\mu = 0$ and standard deviation $\sigma = 1$ was added to each dataset. The code and datasets used in these experiments are publicly available at: https://github.com/abdhmohammadi/GLSTSVM-CIL.

Fig. 3 illustrates the results. The top-left figure shows the Accuracy, the top-right shows F1-Score, the bottom-left shows G-Mean, and the bottom-right represents the average of the three metrics across all imbalance ratios.

Overall, Proposed Method achieved superior or comparable performance in most cases. The integration of strengths from both LS-ATWSVM and LSFLSTSVM-CIL contributes to its robustness, especially in terms of average performance.

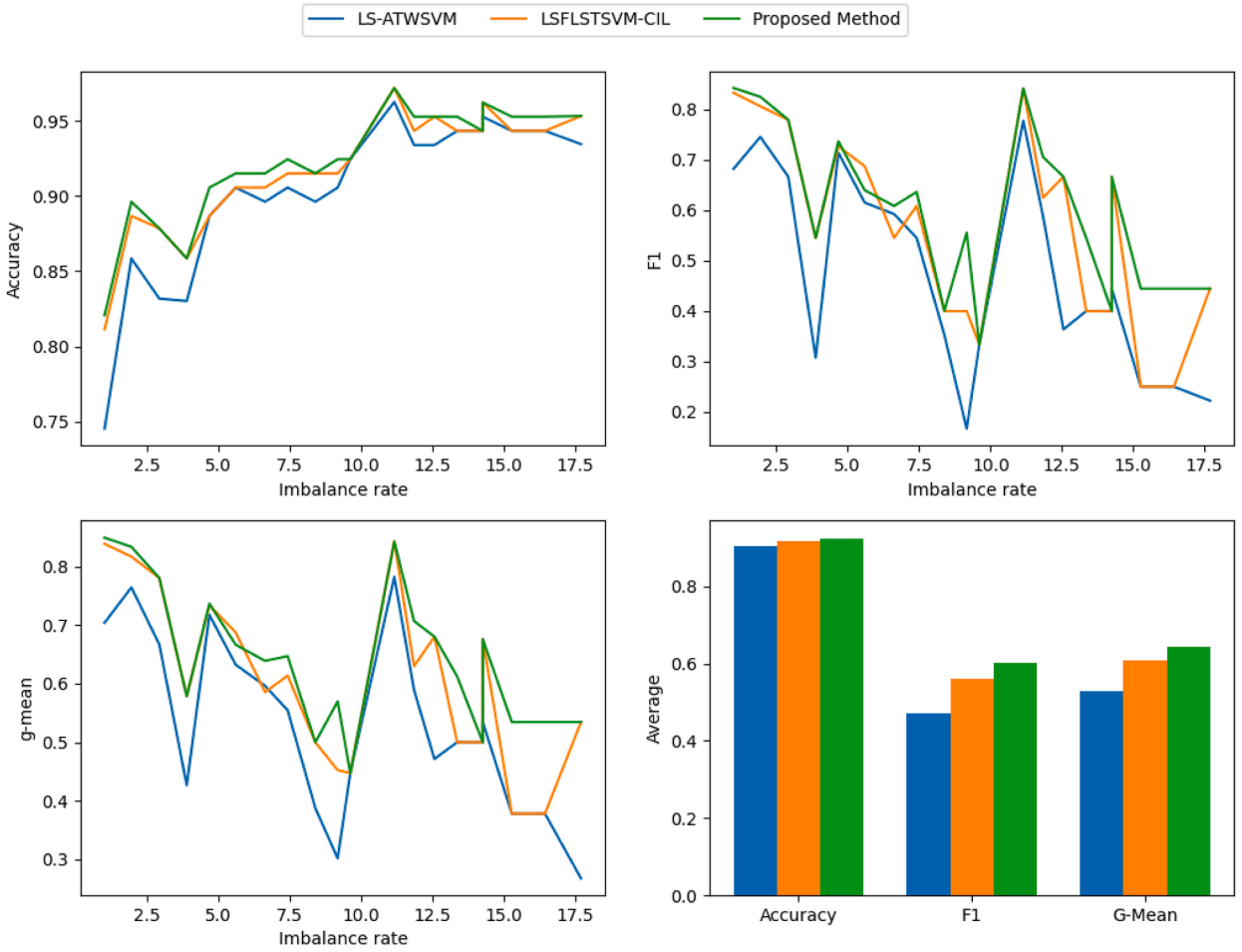

Fig. 3. Noisy data with imbalance rate evaluation in LS-ATWSVM, LSFLSTSVM-CIL, GLSTSVM-CIL.

However, in certain imbalance ratios, particularly when the minority class is extremely underrepresented (e.g., imbalance rates > 15), Proposed Method shows relatively lower F1-Score and G-Mean. This can be attributed to two factors: (i) the extreme scarcity of positive samples causes insufficient support vectors in the minority class, and (ii) although our model uses dual-weighting, its objective function still leans slightly toward maximizing overall margin, which may favor the majority class. These issues are common in margin-based classifiers under severe imbalance, and future enhancements may include adaptive resampling or hybrid loss functions to mitigate this.

The comparison also shows that the weighting strategy in LSFLSTSVM-CIL is more effective than LS-ATWSVM in highly imbalanced scenarios, and our dual weighting in both objective and constraints has further improved the balance between sensitivity and specificity.

5.2. Experiments on class imbalance data

To evaluate the performance of Proposed Method in class inbalance data, 26 datasets were selected from the KEEL repository and were compared with FTSVM [6], LSTSVM [4], LS-ATWSVM [5], SVM-Based SA-Method [8] and LSFLSTSVM-CIL [10] using two criteria, Accuracy and F1-Score. First, the parameters were selected using the 5-Folds and GridSearch for the highest Accuracy and the corresponding F1-Score. For calculating the weight of all points in the Gravity function, the search radius to find number of neighboring points was considered as $r = 0.1$. Then the data was randomly divided into training and test with a ratio of 70-30. Table 10 shows the details of this comparison (In this table and the following tables, the underlined numbers in each row indicate the highest score in that row). the table shows Proposed Method has obtained a better average with distance 0.63 from LSFLSTSVM-CIL and 0.58 from SVM-Based SA-Method, the average score difference in the F1-Score shows a better values, so that Proposed Method has performed with distance 1.1345 approximately better than both methods. The comparison of the average ranks shows

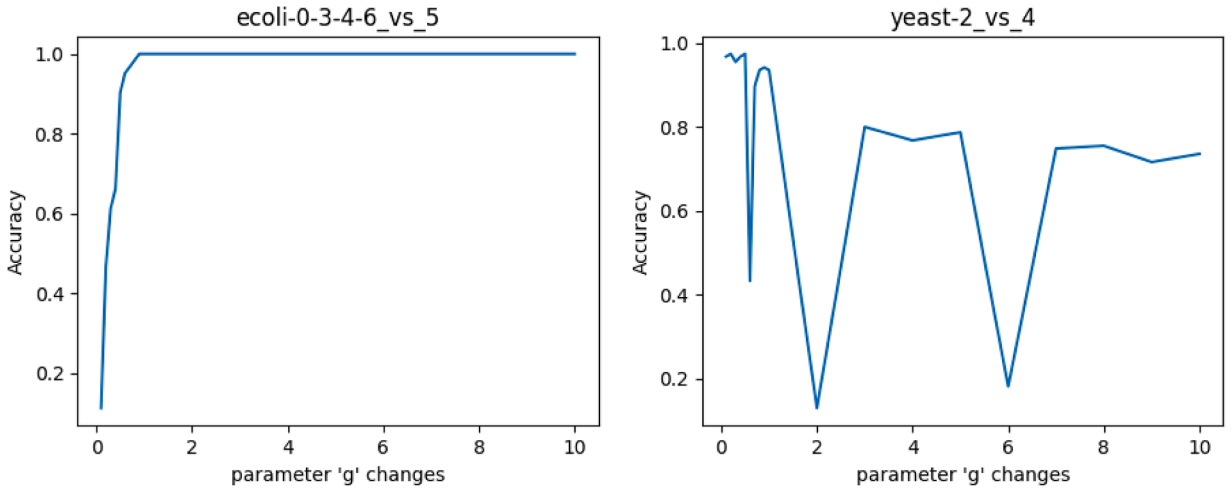

Fig. 4. Effect of parameter g on yeast-2_vs_4 and ecoli-0-3 4-6_vs_5.

FTSVM, LSTSVM and LS-ATSVM have obtained lower ranks in both criteria. Comparing the number of wins and losses confirms this situation well. Proposed Method with 9 wins and LSFLSTSVM-CIL and SVM-Based SA-method with 5 wins have obtained the first and second place, respectively. But this situation in F1-Score is 12, 2 and 5 respectively in favor of Proposed Method. In F1-Score, the *SVM-Based SA-Method* outperformed *LSFLSTSVM-CIL* by 5 wins. The comparison between Accuracy and F1-Score shows that the performance of Proposed Method in F1-Score was much better than the others, and this indicates that the research objective, which is to achieve a better diagnosis of the smaller class, has been well achieved. Fig. 9 shows the overall execution time of the algorithms using Table 10. GLSTSVM-CIL calculates weight values for all points in the dataset, which imposes a lot of calculation time. This situation also occurs in FTSVM. In this respect, tow algorithms perform similarly. However, Proposed Method performed better than FTSVM in 20 out of 26 datasets. In this respect, it did not performe better than other methods.

In the Fig. 4, the changes of the parameter g on the datasets **yeast-2_vs_4** and **ecoli-0-3 4-6_vs_5** is shown. as expected, the effect of this parameter is different according characteristics of the datasets. In **ecoli-0-3-4-6_vs_5**, it goes almost smoothly and reaches the appropriate Accuracy at values higher than 1, but in **yeast- 2_vs_4** is observed in a vacillatory form and results in better Accuracy in values less than 1. Next, in the Fig. 5, mutual changes of two parameters c_2 and c_3 on these datasets is visible.

5.3. Experiments on noisy data

Classification of noisy data is one of the important challenges in machine learning. In this section, in order to check the performance of Proposed Method in noisy data, the Table 11 contains 30 datasets with noise of 10 %, 15 % and 20 %. The datasets are available in KEEL repository. The results show very little difference in the performance of the methods, FTSVM and LSTSVM are in the second ranks with a small distance. Considering the closeness of the ranks, attention to detail seems useful. Proposed Method has a weak performance in the **pima-20an** and **pima-20cn** in F1-Score. And it shows average performance in Accuracy. In the classification of **heart-20cn**, although it is in the second place, it has a big difference with the first place. But in **heart-10an** and **heart-10cn** it has won the first rank with a great margin. This difference is even better in F1-Score. Also, in **pima-15cn**, by obtaining $F1 = 0.6580$, it has a good distance from others. The results of Table 11 are also interesting in the number of wins and losses, in the number of wins, the highest points are obtained by Proposed Method and LSFLSTSVM-CIL. belongs, in this regard, the Table 12 has been prepared according to the results of the Table 11. In this table, it can be seen that Proposed Method has optained second rank of losses, but its winning percentage is significantly better than other methods, so that in 26.67 % cases in terms of Accuracy it has won and has had the highest F1-Score in 40 % cases. As a result of this discussion, it can be said that Proposed Method has performed better than other methods with a very small difference in the average and rank and also according to the amount of wins and losses.

5.4. Experiments on NDC data

To evaluate Proposed Method's performance on large datasets, the NDC (Normal Distributed Clusters) dataset [37] was utilized. Experiments were conducted on a machine with specifications detailed in Table 3 at the computer laboratory of the Faculty of Mathematical Sciences, Ferdowsi University of Mashhad.

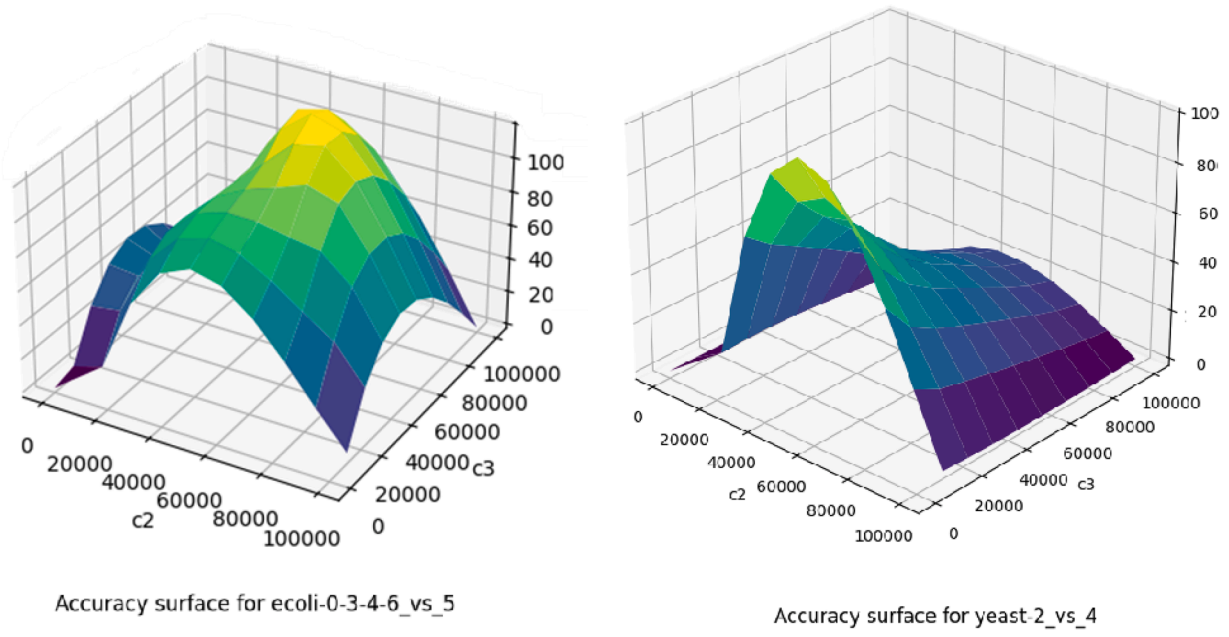

Fig. 5. Effect of parameters c_2 and c_3 on yeast-2_vs_4 and ecoli-0-3 4-6_vs_5.

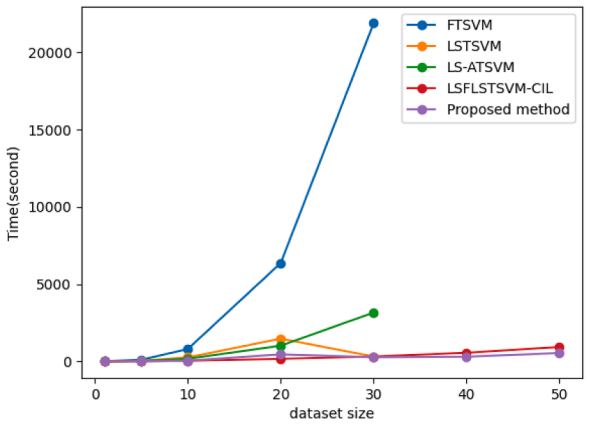

Fig. 6. Comparison of train time in LS-ATWSVM, LSFTSVM-CIL, FTSVM, LSTSVM and Proposed Method on NDC datasets.

Table 14 summarizes the experimental results. The hyperparameters of all methods were tuned to achieve 100 % Accuracy and F1-Score. The **RBF** kernel was used for all models. The reported running time for each method consists of two components: *weight matrix computation time* and *model training time*. LSTSVM and LS-ATWSVM, which do not use a weight matrix, have zero weight computation time.

FTSVM, LSTSVM, and LS-ATWSVM were unable to classify datasets exceeding 30K samples. For the 30K dataset, Proposed Method's total time (including weight calculation and training) was approximately 42.64 min, which is less than the time required by FTSVM, LSTSVM, and LS-ATWSVM. While slightly more time-consuming than LSFTSVM-CIL, it's crucial to note that Proposed Method calculates weights for all samples, whereas LSFTSVM-CIL only processes negative class samples. Furthermore, Except for the weight calculation time, the training time of Proposed Method is superior to LSFTSVM-CIL in 5 of 7 datasets and is almost equal for the 10K. The observed longer training time for 20K could be cautiously attributed to an uncontrolled event by the operating system or hardware. The proposed model successfully classifies datasets up to **50K in size**. Excluding weight calculation time, Proposed Method generally exhibits lower training times compared to other methods, with the 20K dataset being an exception. This highlights the model's **scalability and robustness** in handling large-scale data.

Fig. 7 compares the weight calculation time and training time for Proposed Method. The most time is consumed during the "weight matrix calculation". This process involves distance calculations and updates for all samples, which

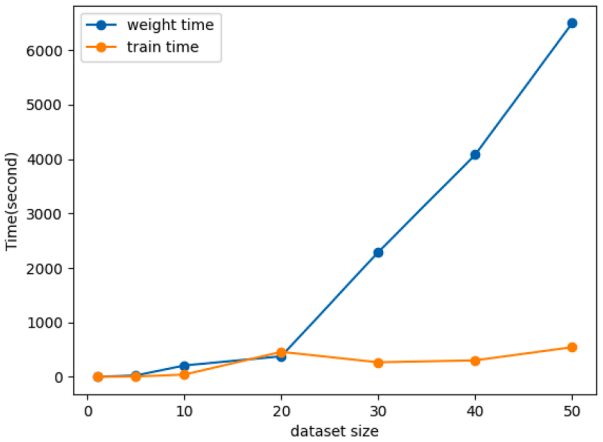

Fig. 7. Comparison of weight time and train time of Proposed Method on NDC datasets.

leads to an increase in computational cost related to the size of the dataset. Fig. 6 also provides a comparative view of the "training time" in different models. Despite the higher computational time imposed by calculating the weight of the entire sample, in terms of "training time", Proposed Method can be said to be no different from "LSFLSTSVM-CIL" - We performed the weight calculations here with global search, future works could explore reducing this overhead using more advanced search methods, approximation strategies, or parallel processing.

5.5. Statistical analysis

Here we presented two statistical analysis to support Tables 10 and 11: *Friedman* test and *Nemenyi Post Hoc*, Each of which is explained below.

5.5.1. Friedman test

In this section, we provide a short statistical analysis for both Accuracy and F1-Score criteria. One of the tests used in this field is the Friedman test. The Friedman test is a non-parametric test that is used to compare and rank the average scores of different groups. The null hypothesis in this test is based on the equality of the means, and the rejection of the null hypothesis means that at least two groups are significant difference together [38]. Considering K algorithms to be compared, which are scored by N datasets. If $\bar{R}_j$ is the average score of jth algorithm, Friedman's statistic using χ^2 distribution with $(K-1)$ degrees of freedom, is defined as $\chi_F^2 = \frac{12N}{K(K+1)}\left(\sum_{j=1}^{K} \bar{R}_j^{\,2} - \frac{K(K+1)^2}{4}\right)$. It is proven that this statistic works unfavorably conservative [39]. Therefore, based on this distribution, the statistic F_F is made as $F_F = \frac{(N-1)\chi_F^2}{N(K-1)-\chi_F^2}$ which has $K-1$ and $(K-1)(N-1)$ degrees of freedom.

Table 10: According to Table 10 and considering $K = 6$ and $N = 26$ in order to analyze the Accuracy, the values of $\chi_F^2 \approx 17.2802$ and $F_F \approx 3.8326$ are obtained. On the other hand, F_F has the distribution F with the degree of freedom $(5, 125)$. The Friedman test results demonstrate statistically significant differences among methods at all conventional significance levels, as the test statistic exceeds critical values for $\alpha = 0.01$ (3.1671), $\alpha = 0.05$ (2.2868), and $\alpha = 0.10$ (1.8939). This provides strong evidence to reject the null hypothesis that all algorithms have identical effects, confirming that at least one method differs significantly from the others in the ranked outcomes across datasets. The findings indicate observed variations are not attributable to random chance.

Table 11: Regarding the Accuracy values reported in Table 11, the Friedman test yields $\chi_F^2 = 8.0851$ and $F_F = 2.0951$. Given that 30 datasets were used, the degrees of freedom are $(4, 116)$. The critical values of the F-distribution $F(4, 116)$ at significance levels $\alpha = 0.01$, 0.05, and 0.10 are 3.4852, 2.4499, and 1.9940, respectively. Since the observed F_F is greater than the critical value at $\alpha = 0.10$, but less than the thresholds for $\alpha = 0.05$ and $\alpha = 0.01$, the null hypothesis is rejected only at the 10 % significance level. This implies that with 90 % confidence, there is a statistically significant difference in the performance ranks of the methods in terms of Accuracy. However, the evidence is not strong enough to support rejection at the more stringent 5 % or 1 % levels, suggesting that while differences exist, they are not highly pronounced.

5.5.2. Nemenyi Post-Hoc test

Addition to Friedman test, has been employed the Nemenyi post-hoc analysis [40,41] for rigorous comparison. This non-parametric approach is recommended for classifier comparisons. it is used when all methods are compared to each

Table 5
Critical Distances (CD) for different α devels.

α	$q_{\alpha,5}$	$q_{\alpha,6}$	CD(Noisy)	CD(Imbalance)
0.05	2.728	2.850	1.196	1.376
0.10	2.459	2.589	1.078	1.251

Table 6
Nemenyi Post-Hoc pairwise comparison for **Imbalance** datasets (**Accuracy**).

Comparison	Rank Difference	$\alpha = 0.05$	$\alpha = 0.10$
Proposed vs. LS-ATWSVM	1.635	✓	✓
Proposed vs. FTSVM	1.404	✓	✓
Proposed vs. LSFLSTSVM-CIL	0.635	–	–
Proposed vs. LSTSVM	1.635	✓	✓
Proposed vs. SVM-Based SA-Method	0.577	–	–

Table 7
Nemenyi Post-Hoc pairwise comparison for **Imbalance** datasets (**F1-Score**).

Comparison	Rank Difference	$\alpha = 0.05$	$\alpha = 0.10$
Proposed vs. LS-ATWSVM	2.250	✓	✓
Proposed vs. FTSVM	1.866	✓	✓
Proposed vs. LSFLSTSVM-CIL	**1.135**	–	–
Proposed vs. LSTSVM	2.058	✓	✓
Proposed vs. SVM-Based SA-Method	**1.116**	–	–

Table 8
Nemenyi Post-Hoc pairwise comparison for **Noisy** datasets (**Accuracy**).

Comparison	Rank Difference	$\alpha = 0.05$	$\alpha = 0.10$
Proposed vs. LS-ATWSVM	0.799	–	–
Proposed vs. FTSVM	0.883	–	–
Proposed vs. LSFLSTSVM-CIL	0.850	–	–
Proposed vs. LSTSVM	0.483	–	–

other. Post-hoc analysis uses the Nemenyi test with Critical Difference (CD):

$$\mathrm{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{66}$$

Considering Table 5 in [40] for q_α, and Tables 10 and 11 with $k = 5, 6$ methods, critical distances where calculated at $\alpha = 0.05$ and 0.10. Table 5 shows these values, where $q_{\alpha,6}$ is q-values for *Imbalanced* datasets in Table 10 and $q_{\alpha,5}$ is for *Noisy* datasets in Table 11. pairwise differences in mean ranks between methods were calculated in Tables 6–9. Classifiers with Rank Differenceerences exceeding CD are significantly different.

Table 10: Proposed Method achieved the highest overall ranking (Accuracy: 2.519 and F1-Score:2.096), According to the observations in Tables 6 and 7, the proposed method has a significant difference with FTSVM, LSTSVM and LS-ATWSVM. In Table 6, there is no significant difference between the proposed method and SVM-Based SA-Method and LSFLSTSVM-CIL. It seems that all three methods have similar performance in terms of overall accuracy. Table 7 shows the CD value against SVM-Based SA-Method and LSFLSTSVM-CIL very close to the critical region. This can be interpreted as a signal of the superiority of the proposed method over these two methods. Although the overall accuracy between these methods is very low (approximately 0.6), this value reaches more than 1.1 in determining the F1-Score. The distance of this value from the critical region is only 0.2 which indicates the superiority of the proposed method.

Table 11: Pairwise comparisons between the classifiers using average ranks of Noisy datasets for Accuracy and F1-Score are shown in Tables 8 and 9. In Table 8, there is no significant difference between the methods in determining the overall accuracy. All methods perform almost the same. The accuracy loss in LSFTSVM-CIL, FTSVM and LS-ATSVM is greater than that of LSTSVM. The results of Table 9 show that LS-ATWSVM has shown more weakness than the other methods compared to the proposed method. No significant difference can be observed in relation to other methods.

Table 9
Nemenyi Post-Hoc pairwise comparison for **Noisy** datasets (**F1-Score**).

Comparison	Rank Difference	$\alpha = 0.05$	$\alpha = 0.10$
Proposed vs. LS-ATWSVM	**1.076**	–	–
Proposed vs. FTSVM	0.483	–	–
Proposed vs. LSFLSTSVM-CIL	0.300	–	–
Proposed vs. LSTSVM	0.216	–	–

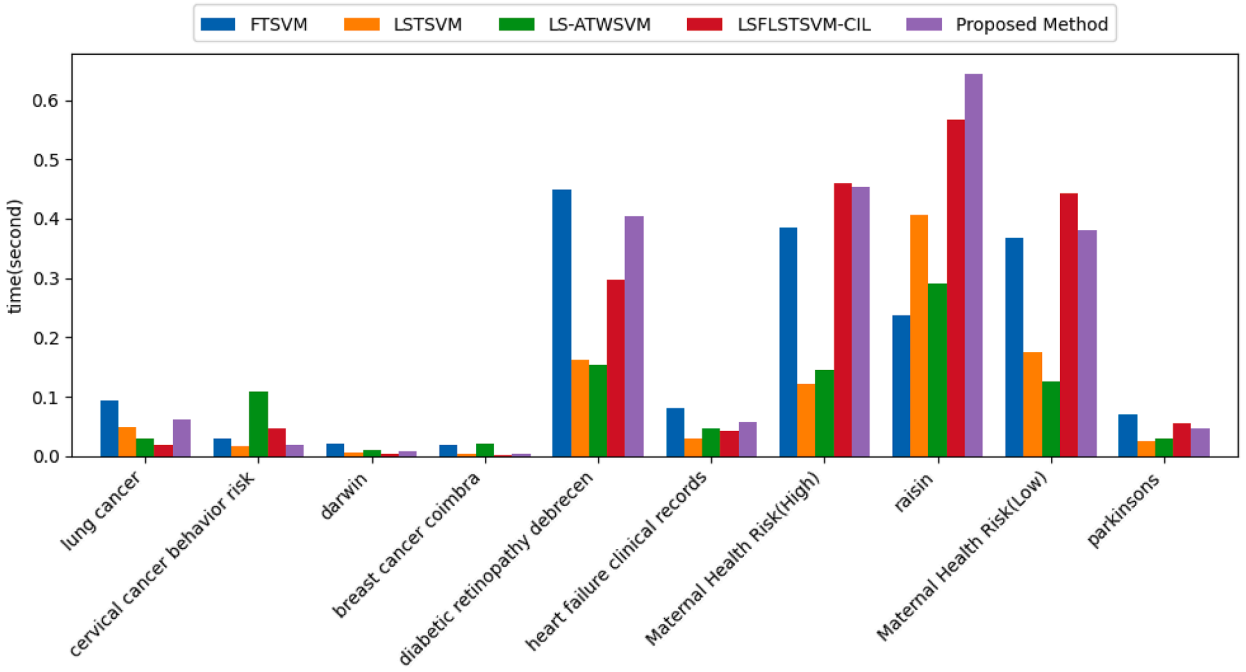

Fig. 8. Comparison of total execution time (weighting time + training time) on medical datasets(Table 15).

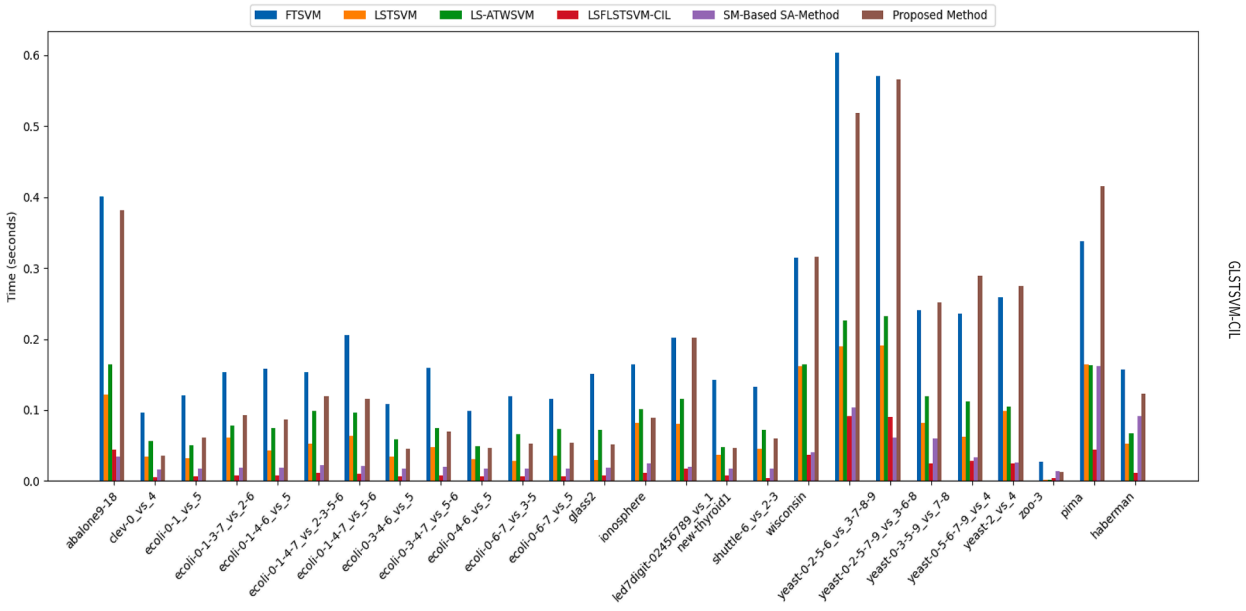

Fig. 9. Comparison of total execution time on imbalance datasets (Table 10).

6. Applications of proposed method

In this section, we present the result of applying Proposed Method to two practical case studies: medical data classification and text data classification.

6.1. Medicine and health data

The Table 15 shows the details of the results in different fields of medical data including 10 datasets from the UCI repository, which include Cancer diseases, Diabetes, Heart diseases, Parkinson's and etc. The search radius of neighborhood points in the weight function for all datasets fixed to $r = 0.1$. The evaluations were done non-linearly with the Gaussian kernel function. According to the table in the dataset of Darwin and Cervical cancer behavior risk, all methods have obtained the same results. The biggest difference in F1-Score between Proposed Method and the previous ranking is observed in the "Breast cancer coimbra" equal to 15.68 %. The comparison of the average results shows that Proposed Method has a distance of approximately 2 % from FTSVM in the Accuracy and with a difference of approximately 3 % of LSFLSTSVM-CIL has taken the first place in F1-Score. The ranking criteria and the number of wins and losses also show its superiority over other methods. It has obtained the best results in the Accuracy criterion with 4 wins and in the F1 criterion with 5 wins and no losses. FTSVM and LSFLSTSVM-CIL rank second in the Accuracy and F1-Score, respectively. Proposed Method did not get the weakest score in the classification of any of the datasets. Like the previous tables, this table also shows a better performance in obtaining F1-Score.

In Fig. 8, we have made a brief comparison based on the execution time of the models using the Table 15. The results are not far from the expected, in fact, a similar pattern is observed to the results in Table 10. In this case, Proposed Method also is better than FTSVM in 7 of 10 cases. LSFLSTSVM-CIL performs weight calculations only on negative class, and the LSTSVM and LS-ATWSVM do not have a weight function. However, Proposed Method was better than LSFLSTSVM-CIL in 4 of 10 cases. Although the result of time is not good, it shows that Proposed Method has good potential for improvement through weight function optimization, which can be considered in future work.

6.2. Text classification

In Table 13, six text datasets including spam messages, sentiments analysis of social network users, analysis of news and sports articles have been reviewed. All sets except the "Sports articles for objectivity analysis" have been transformed into vectors by the TF-IDF method [42]. The sigmoid kernel function has been used to train the models. Also, the neighborhood radius for the Gravity function(defined in Eq. (14)) in these datasets is variable, which is given in the list of parameters. It can be seen that in the "Spam text message" dataset, the best result is obtained with the Accuracy of 100%. In the classification of "Twitter messages", it has the same score as LS FLSTSVM-CIL, but in the same dataset, it has performed better than all methods in terms of F1-Score. Although LSFLSTSCM-CIL is equal to Proposed Method in terms of Accuracy, it shows a lower value in F1-Score. The distance of F1-Score is very significant in the "Redit Comments" and "Twitter messages". In these datasets, it has obtained the best result with a difference of approximately 10 %. The overall results of this table in the average and rank show that Proposed Method performed better compared to other methods, especially the results of F1 clearly show its ability to detect the smaller group. Similar to the results of Table 10, it has performed better than FTSVM and LSFLSTSVM-CIL in the Accuracy with 2.45 % and F1-Score with 7.81 %.

7. Discussion of results

The experimental findings highlight both the strengths and limitations of the proposed GLSTSVM-CIL method. The gravitational weighting mechanism significantly enhanced minority-class recognition, reflected in F1-Score improvements across a wide range of imbalance scenarios. By combining imbalance-ratio scaling and local instance density into a dynamic weighting scheme, GLSTSVM-CIL effectively reduced bias toward the majority class. This advantage was particularly notable at high imbalance ratios, where traditional models typically degrade.

In noisy environments evaluated on 30 benchmark datasets with 10, 15 and 20 % of noise. GLSTSVM-CIL in F1-Score has won almost 40 % of cases (12 wins out of 29 in Table 11). This robustness appears to stem from the inherent properties of its gravitational function, which naturally suppresses the influence of isolated or noisy instances due to their low density and large centroidal distance.

Geometric insights further confirmed the model's efficacy. Visualization of decision boundaries (Fig. 2) revealed that angular optimization led to improved class separation with hyperplane angle of 40.41° compared to 18.82–23.68° in baseline methods. Moreover, the Sum of Squared Distance (SSD) was significantly reduced (8.55 vs. $>$17.87), suggesting better alignment with complex class distributions.

Table 10
Experiments on class imbalance data.

dataset	FTSVM Accuracy, F1,Time ($c_1 = c_2, \mu$)	LSTSVM Accuracy, F1,Time ($c_1 = c_2, \mu$)	LS-ATWSVM Accuracy, F1,Time ($c_1 = c_5, c_2, c_3, \mu$)	LSFLSTSVM-CIL Accuracy, F1,Time ($c_0, c_1 = c_2, c_3 = c_4, \mu$)	SVM-Based SA-Method Accuracy, F1,Time (c, μ)	Proposed Method Accuracy, F1,Time (c_1, c_2, c_3, μ, g)
abalone9-18	0.9636, 0.6923, 0.4009 (10, 1)	0.9455, 0.4545, 0.1211 (10^3.1)	0.9545, 0.5000, 0.1646 (1, 0.1, 0.1, 1)	0.9545, 0.4444, 0.0444 (0.5, 10^{-5}, 10^{-5}, 10^{-5})	0.9631, 0.6300, 0.0346 (10^3, 1)	**0.9773**, **0.7826**, **0.3817** (10, 10^{-4}, 10^5, 10^{-3}, 0.1)
clev-0_vs_4	0.8846, 0.5000, 0.0957 (10^3, 10^{-4})	0.8846, 0.5000, 0.0344 (10^2, 10^{-4})	0.4231, 0.2105, 0.0563 (1, 10^{-5}, 10^{-5}, 1)	0.9038, 0.4444, 0.0056 (1.5, 10, 10^{-4}, 10^{-4})	0.8845, 0.9373, 0.01658 (10^5, 10^{-3})	0.8462, 0.4286, 0.0359 (1, 0.1, 10^3, 10^{-4}, 1)
ecoli-0-1_vs_5	0.9583, 0.7273, 0.1208 (10^{-2}, 10^{-3})	0.9583, 0.7273, 0.0319 (0.1, 10^{-3})	0.9583, 0.7273, 0.0500 (1, 0.1, 0.1, 1)	0.9583, 0.7273, 0066 (2, 1, 1, 1)	0.9833, 0.8762, 0.0177 (10^5, 10^{-3})	0.9861, 0.9333, 0.0615 (10, 10^{-4}, 10, 10^{-3}, 2)
ecoli-0-1-3-7_vs_2-6	1, 1, 0.1537 (10^{-3}, 10^{-3})	1, 1, 0.0609 (0.1, 10^{-3})	1, 1, 0.0778 (1, 0.1, 0.1, 1)	1, 1, 0.0077 (0.5, 10^5, 10^5, 10^5)	0.9929, 0.7333, 0.0181 (10^3, 10^{-4})	1, 1, 0.0925 (10^{-5}, 10^{-5}, 10^5, 10^{-3}, 0.3)
ecoli-0-1-4-6_vs_5	0.9762, 0.9000, 0.1576 (1, 10^{-4})	0.9643, 0.8421, 0.431 (1, 10^{-4})	0.9762, 0.9091, 0.0750 (1, 0.1, 0.1, 1)	0.9524, 0.7778, 0.0075 (1.5, 10^4, 10^5, 10^5)	0.9821, 0.8444, 0.0189 (10^5, 10^{-3})	0.9881, 0.9524, 0.0.0869 (10, 0.1, 10^4, 10^{-3}, 4)
ecoli-0-1-4-7_vs_2-3-5-6	0.9604, 0.7500, 0.1535 (1, 10^{-4})	0.9703, 0.8000, 0.0524 (0.1, 10^{-3})	0.9703, 0.8000, 0.0982 (1, 0.1, 0.1, 1)	0.9802, 0.8750, 0.0109 (1, 1, 10^{-2}, 10^{-2})	0.9761, 0.8351, 0.0217 (10, 10^{-4})	0.9505, 0.6667, 0.0.1190 (1, 0.1, 10^4, 10^{-3}, 0.8)
ecoli-0-1-4-7_vs_5-6	0.9700, 0.5714, 0.2059 (1, 10^{-4})	0.9600, 0.5000, 0.0629 (1, 10^{-4})	0.9700, 0.5714, 0.0961 (1, 0.1, 0.1, 1)	0.9700, 0.5714, 0.0097 (2.5, 10^4, 10^{-2}, 10^{-2})	0.9608, 0.9786, 0.0215 (10^5, 10^{-5})	0.9800, 0.6667, 0.1161 (10^2, 1, 10^5, 10^{-3}, 1)
ecoli-0-3-4-6_vs_5	0.9677, 0.8333, 0.1085 (10^{-2}, 10^{-3})	0.9677, 0.8333, 0.0345 (1, 10^{-3})	0.9839, 0.9231, 0.0582 (1, 0.1, 0.1, 1)	1, 1, 0.0058 (2.5, 10^5, 10^4, 10^4)	0.9751, 0.8540, 0.0176 (10^2, 10^{-3})	1, 1, 0.0453 (10, 10^{-2}, 10^2, 10^{-3}, 10)
ecoli-0-3-4-7_vs_5-6	0.9487, 0.8182, 0.1591 (1, 10^{-4})	0.9103, 0.6316, 0.0476 (0.1, 10^{-3})	0.9359, 0.8000, 0.0746 (1, 0.1, 0.1, 1)	0.9615, 0.8696, 0.0083 (0.5, 10, 10^{-2}, 10^{-2})	0.9768, 0.8667, 0.01923 (10^2, 10^{-5})	0.9615, 0.8696, 0.0689 (10^2, 1, 10^5, 10^{-3}, 5)
ecoli-0-4-6_vs_5	0.9672, 0.8000, 0992 (10^{-2}, 10^{-3})	0.9836, 0.9091, 0.0313 (0.1, 10^{-4})	0.9836, 0.9091, 0.4949 (1, 0.1, 0.1, 1)	0.9836, 0.9091, 0.0059 (1.5, 10^4, 10^5, 10^5)	0.9755, 08540, 0.0171 (10, 10^{-4})	1, 1, 0.0461 (10, 10^5, 1, 10^{-3}, 0.5)
ecoli-0-6-7_vs_3-5	0.9104, 0.6667, 1194 (1, 10^{-4})	0.9403, 0.7143, 0.0285 (1, 10^{-4})	0.9254, 0.7059, 0.0660 (1, 0.1, 0.1, 1)	0.9552, 0.8000, 0.0064 (2, 10^5, 0.1, 0.1)	0.9416, 0.9687, 0.0173 (10, 10^{-5})	0.9254, 0.7059, 0.0525 (10, 0.1, 10^3, 10^{-3}, 3)
ecoli-0-6-7_vs_5	0.9697, 0.8571, 0.1156 (1, 10^{-4})	0.9697, 0.8333, 0.0812 (1, 10^{-4})	0.9394, 0.7500, 0.7273 (1, 0.1, 0.1, 1)	0.9848, 0.9231, 0.0065 (1.5, 10, 0.1, 0.1)	0.9773, 0.8548, 0.0173 (10, 10^{-4})	0.9697, 0.8571, 0.0539 (10, 0.1, 10^5, 10^{-3}, 2)
glass2	0.8615, 0.1818, 0.1504 (10^{-2}, 10)	0.8769, 0.3333, 0.0805 (0.1, 10)	0.8615, 0.3077, 0.7185 (1, 10^{-2}, 10^{-2}, 1)	0.7846, 0.2222, 0.0077 (0.5, 10^{-3}, 10^{-5}, 10^{-5})	0.9157, 0.3000, 0.0190 (10, 10)	0.8308, 0.4211, 0.0513 (10, 10^{-4}, 0.1, 0.1, 1)
ionosphere	0.9434, 0.9118, 0.1643 (0.1, 0.1)	0.9434, 0.9118, 0.0373 (0.1, 0.1)	0.934, 0.8955, 0.1005 (1, 0.1, 0.1, 1)	0.9434, 0.9118, 0.0116 (0.5, 1, 10^{-2}, 10^{-2})	0.9787, 0.8548, 0.0248 (10, 0.1)	0.9528, 0.9275, 0.0894 (1, 10^{-2}, 1, 0.1, 10)
led7digit-02456789_vs_1	0.9624, 0.8276, 0.2019 (1, 0.1)	0.9774, 0.8800, 0.0805 (1, 0.1)	0.9624, 0.7826, 0.1151 (1, 0.1, 0.1, 1)	0.9699, 0.8571, 0.0179 (2, 10, 0.1, 0.1)	0.9684, 0.8140, 0.0203 (10^2, 0.1)	0.9850, 0.9231, 0.2017 (10^2, 10^{-5}, 1, 10^{-2}, 2)
new-thyroid1	1, 1, 0.1424 (0.1, 10^{-2})	1, 1, 0.0373 (1, 10^{-2})	0.9846, 0.9565, 0.0471 (1, 0.1, 0.1, 1)	1, 1, 0.0076 (0.5, 10, 10^{-3}, 10^{-3})	0.9953, 0.9867, 0.0174 (10^2, 10^{-3})	1, 1, 0.0462 (0.1, 10^2, 10^{-4}, 10^{-2}, 10)
shuttle-6_vs_2-3	0.9855, 0.8889, 0.1320 (10^{-4}, 10^{-5})	0.9855, 0.9091, 0.0452 (1, 10^{-5})	1, 1, 0.0722 (1, 10^{-2}, 10^{-2}, 1)	1, 1, 0.0043 (0.5, 10^{-5}, 10^{-5}, 10^{-5})	1, 1, 0.0169 (1, 10^{-5})	1, 1, 0.0604 (10^{-3}, 10^{-5}, 10^{-5}, 10^{-5}, 10)
wisconsin	0.9659, 0.9536, 0.3141 (10^3, 10^{-5})	0.9610, 0.9467, 0.1618 (10^4, 10^{-4})	0.9561, 0.9396, 0.1636 (1, 10^{-4}, 10^{-4}, 1)	0.9415, 0.9268, 0.0371 (2.5, 10^{-5}, 10, 10)	0.9692, 0.9561, 0.0401 (10^2, 10^{-3})	0.9756, 0.9673, 0.3162 (10^3, 10^{-3}, 10^5, 0.1, 3)
yeast-0-2-5-6_vs_3-7-8-9	0.9139, 0.5667, 6030 (10, 1)	0.9205, 0.5714, 0.1901 (1, 10)	0.9172, 0.5455, 0.2261 (1, 0.1, 0.1, 1)	0.9205, 0.6250, 0.0912 (0.5, 10, 1, 1)	0.9333, 0.5693, 0.1036 (10, 10)	0.9172, 0.6575, 0.5182 (0.1, 10^{-5}, 10^5, 10^{-4}, 3)
yeast-0-2-5-7-9_vs_3-6-8	0.9636, 0.8254, 0.5702 (0.1, 10)	0.9603, 0.8000, 0.1904 (0.1, 10)	0.9503, 0.7170, 0.2321 (1, 0.1, 0.1, 1)	0.9669, 0.8333, 0.0899 (1, 10, 0.1, 0.1)	0.9382, 0.9657, 0.0615 (10^5, 10)	0.9570, 0.7797, 0.5664 (10^5, 10^2, 10, 10, 8)
yeast-0-3-5-9_vs_7-8	0.8553, 0.2667, 0.2403 (10^2, 10^{-5})	0.8487, 0.2069, 0.0811 (1, 0.1)	0.8487, 0.1481, 0.1192 (1, 10^{-2}, 10^{-2}, 1)	0.8553, 0.2143, 0.0248 (0.5, 10^2, 10, 10)	0.9170, 0.3312, 0.0597 (1, 1)	0.8618, 0.3226, 0.2518 (1, 10^4, 10^{-4}, 1, 4)
yeast-0-5-6-7-9_vs_4	0.8994, 0.4667, 0.2353 (1, 1)	0.8868, 0.3571, 0.0617 (1, 1)	0.9182, 0.6486, 0.1120 (1, 0.1, 0.1, 1)	0.8553, 0.2581, 0.0282 (2, 10, 10^{-5}, 10^{-5})	0.9129, 0.5016, 0.0328 (10^2, 10)	0.9057, 0.5455, 0.2893 (0.1, 10^{-3}, 1, 10^2, 0.2)
yeast-2_vs_4	0.9548, 0.7407, 0.2589 (0.1, 10)	0.9548, 0.7200, 0.0983 (10, 10)	0.9613, 0.7500, 0.1051 (1, 0.1, 0.1, 1)	0.9613, 0.7692, 0.0248 (1, 10^3, 10^2, 10^2)	0.9611, 0.7925, 0.0254 (10^4, 0.1)	0.9742, 0.8462, 0.2752 (10^2, 10^{-3}, 10^4, 10^{-2}, 0.2)
zoo-3	0.9677, 0.6667, 0.0269 (1, 0.1)	0.9677, 0.6667, 0.0013 (0.1, 0.1)	0.9355, 0.5000, 0.0016 (1, 0.1, 10^{-3}, 1)	0.9677, 0.6667, 0.0046 (2, 10^{-5}, 0.1, 0.1)	0.9610, 0.5333, 0.0141 (10^5, 10^{-3})	0.9677, 0.6667, 0.0131 (10^5, 10^5, 10^5, 0.1, 10)
pima	0.7619, 0.6452, 0.3379 (1, 10^{-5})	0.7273, 0.5772, 0.1641 (0.1, 10^{-5})	0.7489, 0.6234, 0.1624 (1, 10^{-2}, 10^{-3}, 1)	0.7489, 0.6848, 0.0445 (1, 10^5, 10^5, 10^5)	0.7396, 0.5903, 0.1619 (10^2, 10^{-4})	0.7532, 0.6851, 0.4148 (10^4, 10^{-4}, 1, 10^{-5}, 0.6)
haberman	0.6957, 0.3000, 0.1571 (1, 10^{-4})	0.6739, 0.1667, 0.0525 (10^{-3}, 10^{-4})	0.7174, 0.3158, 0.0669 (1, 10^{-4}, 0.1, 1)	0.6957, 0.3636, 0.0108 (0.5, 1, 10^{-2}, 10^{-2})	0.6992, 0.3777, 0.0916 (10^4, 10^2)	0.7391, 0.5385, 0.1230 (1, 10^{-4}, 1, 10^{-5}, 0.3)
average	(0.9311, 0.7061)	(0.9284, 0.6843)	(0.9122, 0.6899)	(0.9314, 0.7183)	(0.9058, 0.7503)	(0.9386, 0.7747)
Rank	(3.9231, 3.9615)	(4.1538, 4.1538)	(4.1538, 4.3462)	(3.1538, 3.2308)	(3.0962, 3.2115)	(2.5192, 2.0962)
Accuracy(Win,Tie,Loss)	(1, 16, 3)	(0, 14, 5)	(1, 15, 5)	(5, 16, 4)	(5, 1, 2)	(10, 9, 1)
F1(Win,Tie,Loss)	(0, 9, 4)	(0, 9, 5)	(1, 7, 8)	(2, 10, 4)	(5, 1, 2)	(12, 8, 1)

Table 11
Experiments on noisy data.

dataset	FTSVM Accuracy, F1 $(c_1 = c_2, \mu)$	LSTSVM Accuracy, F1 $(c_1 = c_2, \mu)$	LS-ATWSVM Accuracy, F1 $(c_1 = c_5, c_2, c_3, \mu)$	LSFLSTSVM-CIL Accuracy, F1 $(c_0, c_1 = c_2, c_3 = c_4, \mu)$	Proposed Method Accuracy, F1 $(c_1, c_2, erc_3, \mu, g)$
heart-10an	0.6790, 0.5667 $(1, 10^{-5})$	0.6543, 0.5172 $(10^{-4}, 10^{-5})$	0.6049, 0.3846 $(1, 10^{-3}, 10^{-5}, 1)$	0.6543, 0.5172 $(2, 10^{2}, 1, 10^{-5})$	0.7284, 0.7027 $(1, 10^{3}, 10^{5}, 10^{-5}, 1)$
heart-10cn	0.7284, 0.7800 $(10^{5}, 10^{-5})$	0.7037, 0.7000 $(10^{-4}, 10^{-5})$	0.6420, 0.6234 $(1, 10^{-3}, 10^{-2}, 1)$	0.6914, 0.7059 $(0.5, 1, 0.1, 10^{-5})$	0.7901, 0.8283 $(10^{-3}, 10, 10, 10^{-5}, 1)$
heart-15an	0.8025, 0.8095 $(10^{5}, 10^{-5})$	0.5432, 0.3934 $(10^{-4}, 10^{-5})$	0.5926, 0.5714 $(1, 10^{-3}, 10^{-3}, 1)$	0.6420, 0.6420 $(0.5, 0.1, 10^{-3}, 10^{-3})$	0.7778, 0.8000 $(0.1, 10^{2}, 10^{3}, 10^{-5}, 1)$
heart-15cn	0.6667, 0.6747 $(10, 10^{-4})$	0.6296, 0.5833 $(10^{2}, 10^{-4})$	0.6049, 0.5789 $(1, 10^{-3}, 10^{-3}, 1)$	0.7284, 0.7381 $(2.5, 1, 0.1, 10^{-5})$	0.6914, 0.6479 $(10^{-3}, 10, 10^{2}, 10^{-5}, 1)$
heart-20an	0.6667, 0.6197 $(10^{2}, 10^{-5})$	0.6543, 0.5000 $(10^{-4}, 10^{-5})$	0.6173, 0.5079 $(1, 10^{-3}, 10^{-3}, 1)$	0.6543, 0.5882 $(1, 1, 10^{-2}, 10^{-5})$	0.6914, 0.6835 $(0.1, 10^{2}, 10^{4}, 10^{-5}, 1)$
heart-20cn	0.6049, 0.6444 $(1, 10^{-4})$	0.5802, 0.6304 $(1, 10^{-4})$	0.4568, 0.3889 $(1, 10^{-2}, 10^{-2}, 1)$	0.7160, 0.8067 $(2, 1, 0.1, 10^{-5})$	0.6667, 0.7097 $(10^{-2}, 10^{2}, 10^{2}, 10^{-5}, 1)$
iono-10an	0.8868, 0.8667 $(0.1, 0.1)$	0.8962, 0.8791 $(1, 0.1)$	0.8962, 0.8736 $(1, 0.1, 0.1, 1)$	0.8868, 0.8537 $(2, 10^{3}, 10, 0.1)$	0.8396, 0.8000 $(10^{-3}, 10^{-3}, 0.1, 0.1, 1)$
iono-10cn	0.8679, 0.8511 $(1, 10^{-2})$	0.8868, 0.8750 $(1, 0.1)$	0.8774, 0.8632 $(1, 0.1, 0.1, 1)$	0.8868, 0.8776 $(2.5, 10^{5}, 10^{3}, 0.1)$	0.8868, 0.8846 $(10^{-4}, 10^{-5}, 10^{-3}, 0.1, 1)$
iono-15an	0.8491, 0.7714 $(0.1, 0.1)$	0.8868, 0.8378 $(1, 0.1)$	0.8962, 0.8406 $(1, 0.1, 0.1, 1)$	0.8962, 0.8493 $(1, 0.1, 10^{-2}, 0.1)$	0.8396, 0.7385 $(0.1, 0.1, 10, 0.1, 1)$
iono-15cn	0.8962, 0.8791 $(1, 10^{-2})$	0.8679, 0.8478 $(1, 0.1)$	0.8679, 0.8409 $(1, 0.1, 0.1, 1)$	0.9151, 0.9053 $(2.5, 10^{5}, 10^{3}, 0.1)$	0.8962, 0.8866 $(0.1, 0.1, 0.1, 0.1, 1)$
iono-20an	0.8396, 0.7606 $(0.1, 0.1)$	0.8679, 0.8056 $(1, 0.1)$	0.8491, 0.7838 $(1, 0.1, 0.1, 1)$	0.8679, 0.8056 $(0.5, 0.1, 10^{-2}, 0.1)$	0.8585, 0.7458 $(0.1, 0.1, 10^{5}, 0.1, 1)$
iono-20cn	0.783, 0.6933 $(1, 10^{-2})$	0.8113, 0.7561 $(1, 0.1)$	0.8208, 0.7654 $(1, 0.1, 0.1, 1)$	0.8491, 0.8095 $(1.5, 10^{5}, 10^{3}, 0.1)$	0.8208, 0.7711 $(10^{-3}, 10^{-3}, 10^{-3}, 0.1, 1)$
sonar-10an	0.7937, 0.7547 $(10^{-2}, 1)$	0.8413, 0.7826 $(0.1, 1)$	0.7619, 0.6939 $(1, 0.1, 0.1, 1)$	0.7937, 0.7797 $(2, 10, 1, 1)$	0.8413, 0.8000 $(10, 0.1, 10^{2}, 1, 1)$
sonar-10cn	0.7778, 0.7812 $(0.1, 1)$	0.8095, 0.8182 $(1, 1)$	0.7937, 0.7937 $(1, 0.1, 0.1, 1)$	0.8254, 0.8406 $(2, 0.1, 10^{-2}, 1)$	0.8254, 0.8451 $(1, 10^{5}, 10^{2}, 0.1, 1)$
sonar-15an	0.8413, 0.8148 $(1, 1)$	0.8571, 0.8302 $(1, 1)$	0.7778, 0.6957 $(1, 0.1, 0.1, 1)$	0.8254, 0.7843 $(0.5, 10^{-4}, 10^{-5}, 1)$	0.7937, 0.7547 $(10, 0.1, 10^{3}, 1, 1)$
sonar-15cn	0.7778, 0.7742 $(10, 1)$	0.8413, 0.8438 $(1, 1)$	0.7619, 0.7273 $(1, 0.1, 0.1, 1)$	0.7778, 0.7812 $(1, 10^{-4}, 10^{-5}, 1)$	0.8413, 0.8485 $(0.1, 1, 1, 0.1, 1)$
sonar-20an	0.619, 0.625 $(0.1, 1)$	0.619, 0.6364 $(1, 1)$	0.6349, 0.549 $(1, 0.1, 0.1, 1)$	0.6825, 0.7143 $(2, 10, 1, 1)$	0.6032, 0.6269 $(10, 0.1, 10^{2}, 1, 1)$
sonar-20cn	0.8254, 0.8000 $(0.1, 1)$	0.8254, 0.8000 $(1, 1)$	0.8254, 0.7925 $(1, 0.1, 0.1, 1)$	0.8254, 0.8070 $(0.5, 10^{3}, 10^{-2}, 1)$	0.8571, 0.8421 $(10^{3}, 10^{3}, 10^{5}, 1, 1)$
pima-10an	0.6883, 0.5263 $(0.1, 10^{-4})$	0.7013, 0.4964 $(1, 10^{-5})$	0.7273, 0.4615 $(1, 10^{-2}, 10^{-2}, 1)$	0.5152, 0.3563 $(0.5, 10^{4}, 1, 10^{-5})$	0.7100, 0.4071 $(1, 1, 1, 10^{-5}, 0.3)$
pima-10cn	0.7100, 0.5939 $(1, 10^{-5})$	0.7056, 0.5952 $(0.1, 10^{-5})$	0.7013, 0.5175 $(1, 10^{-2}, 10^{-2}, 1)$	0.5584, 0.5405 $(2.5, 1, 0.1, 10^{-5})$	0.7143, 0.5600 $(1, 10^{-3}, 0.1, 10^{-5}, 0.3)$
pima-15an	0.6494, 0.3721 $(1, 10^{-5})$	0.6623, 0.3810 $(1, 10^{-5})$	0.6797, 0.3729 $(1, 10^{-3}, 10^{-3}, 1)$	0.6061, 0.3636 $(1, 10^{2}, 1, 10^{-3})$	0.6580, 0.3248 $(10^{5}, 10^{4}, 10^{4}, 10^{-5}, 1)$
pima-15cn	0.6840, 0.5876 $(1, 10^{-5})$	0.6883, 0.5909 $(1, 10^{-5})$	0.6537, 0.4937 $(1, 10^{-2}, 10^{-2}, 1)$	0.5974, 0.4804 $(0.5, 0.1, 10^{5}, 10^{-4})$	0.6580, 0.6580 $(0.1, 10^{-3}, 10^{-2}, 10^{-5}, 0.5)$
pima-20an	0.6883, 0.4545 $(1, 10^{-5})$	0.671, 0.4154 $(10.0, 10^{-5})$	0.6970, 0.4167 $(1, 10^{-4}, 10^{-4}, 1)$	0.6277, 0.5426 $(0.5, 0.1, 10^{5}, 10^{-4})$	0.6970, 0.3137 $(10^{-2}, 10^{-5}, 10^{-5}, 10^{-5}, 0.8)$
pima-20cn	0.6970, 0.5882 $(1, 10^{-5})$	0.7100, 0.6298 $(1, 10^{-5})$	0.6364, 0.4474 $(1, 10^{-3}, 10^{-3}, 1)$	0.6623, 0.6355 $(0.5, 0.1, 10^{5}, 10^{-4})$	0.6407, 0.4029 $(10^{-4}, 10^{-5}, 10^{-4}, 10^{-5}, 0.1)$
wdbc-10an	0.8187, 0.7634 $(1, 10^{-4})$	0.8421, 0.8000 $(1, 10^{-4})$	0.8480, 0.7937 $(1, 10^{-2}, 10^{-2}, 1)$	0.6550, 0.6704 $(0.5, 0.1, 10^{5}, 10^{-4})$	0.8596, 0.8033 $(0.1, 0.1, 0.1, 10^{-5}, 0.8)$
wdbc-10cn	0.8772, 0.8571 $(1, 10^{-5})$	0.8947, 0.8816 $(1, 10^{-4})$	0.8947, 0.8767 $(1, 0.1, 0.1, 1)$	0.7193, 0.7474 $(0.5, 0.1, 10^{5}, 10^{-4})$	0.8655, 0.8456 $(0.1, 1, 0.1, 10^{-5}, 0.8)$
wdbc-15an	0.8655, 0.7965 $(0.1, 10^{-5})$	0.8596, 0.7857 $(0.1, 10^{-5})$	0.8947, 0.8421 $(1, 0.1, 0.1, 1)$	0.8889, 0.8348 $(0.5, 10^{2}, 10.0, 10^{-5})$	0.8772, 0.8073 $(0.1, 0.1, 10^{-2}, 10^{-5}, 0.6)$
wdbc-15cn	0.8596, 0.8286 $(1, 10^{-4})$	0.8596, 0.8235 $(1, 10^{-5})$	0.8304, 0.7852 $(1, 0.1, 0.1, 1)$	0.4561, 0.6265 $(2, 10^{4}, 10^{3}, 10^{-5})$	0.8655, 0.8414 $(0.1, 0.1, 1, 10^{-5}, 0.2)$
wdbc-20an	0.7719, 0.6723 $(10^{-2}, 10^{-5})$	0.8070, 0.7402 $(1, 10^{-5})$	0.8246, 0.7541 $(1, 0.1, 0.1, 1)$	0.7661, 0.5918 $(0.5, 10^{5}, 10^{2}, 10^{-5})$	0.7953, 0.6789 $(1, 10, 10^{2}, 10^{-5}, 0.1)$
wdbc-20cn	0.8129, 0.7808 $(1, 10^{-4})$	0.8129, 0.7681 $(1, 10^{-4})$	0.8187, 0.7669 $(1, 0.1, 0.1, 1)$	0.8129, 0.7538 $(1.5, 10^{5}, 10^{3}, 10^{-4})$	0.8480, 0.8143 $(1, 10^{-5}, 10^{-3}, 10^{-5}, 0.1)$
Average	(0.7676, 0.7096)	(0.7664, 0.6982)	(0.7496, 0.6601)	(0.7328, 0.6983)	(0.7813, 0.7124)
Rank	(3.250, 3.050)	(2.850, 2.783)	(3.1667, 3.733)	(3.217, 2.867)	(2.367, 2.567)
Accuracy(Win,Tie,Loss)	(1, 8, 4)	(3, 13, 2)	(4, 7, 9)	(5, 11, 9)	(8, 7, 3)
F1(Win,Tie,Loss)	(2, 1, 3)	(5, 3, 3)	(2, 0, 11)	(8, 2, 7)	(12, 0, 6)

Table 12
Win-Loss percentage in the Table 11.

Method	Accuracy wines(%)	Accuracy loss(%)	F1 wines(%)	F1 loss(%)
FTSVM	3.33 %	13.33 %	6.67 %	10 %
LSTSVM	10 %	6.67 %	16.67 %	10 %
LS-ATWSVM	13.33 %	30 %	6.67 %	36.67 %
LSFLSTSVM-CIL	16.67 %	30 %	26.67 %	23.33 %
Proposed Method	26.67 %	10 %	40 %	20 %

Table 13
Experiments on Text data.

Dataset	FTSVM Accuracy, F1 ($c_1 = c_2, \mu$)	LSTSVM Accuracy, F1 ($c_1 = c_2, \mu$)	LS-ATWSVM Accuracy, F1 ($c_1 = c_5, c_2, c_3, \mu$)	LSFLSTSVM-CIL Accuracy, F1 ($c_1 = c_2, c_3 = c_4, \mu, c_0$)	Proposed Method Accuracy, F1 (c_1, c_2, c_3, μ, g, r)
Spam text message	0.9500, 0.7692 (10^{-2}, 1)	0.9333, 0.6667 (0.1, 10)	0.9500, 0.7692 (1, 0.1, 0.1, 1)	0.9667, 0.8571 ($10^{-5}, 10^{-5}, 10^{-5}$, 1, 1)	1, 1 ($10^{-5}, 10^{-5}, 10^{-5}, 10^{-3}$, 10, 0.9)
Twitter messages	0.6500, 0.4324 ($10^2, 10^{-5}$)	0.6667, 0.3333 (10, 10)	0.6167, 0.1481 (1, 1, 1, 1)	0.7333, 0.5556 (10^{-5}, 10, 10^4, 2.5)	0.7667, 0.6500 ($10^{-5}, 10^{-5}, 10^2, 10^2$, 5, 1)
Sports articles for objectivity analysis	0.8067, 0.7364 (1, 10)	0.8067, 0.7041 ($10^{-4}, 10^{-5}$)	0.7200, 0.6818 (1, 0.1, 0.1, 1)	0.7433, 0.5838 (0.1, 0.1, 0.1, 0.5)	0.7867, 0.6596 (10, 0.1, 10^{-2}, 10^{-5}, 0.4, 0.1)
Reddit comments	0.7333, 0.4667 (10, 10^{-5})	0.7667, 0.5625 (0.1, 10)	0.7167, 0.3704 (1, 0.1, 0.1, 1)	0.7333, 0.6190 (10^2, 10^{-3}, 1, 2.5)	0.8000, 0.7273 (10^{-5}, 10^{-2}, 0.1, 10, 10, 0.1)
TDT2	0.9970, 0.9973 (10^{-5}, 10)	0.9985, 0.9987 ($10^{-5}, 10^{-3}$)	0.9732, 0.9755 (0.1, 0.1, 0.1, 0.1)	0.9062, 0.9086 ($10^{-5}, 10^{-5}, 10^{-5}$, 2)	0.9658, 0.9687 ($10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}$, 0.7, 0.1)
CNAE-1	0.9969, 0.987 (10^2, 1)	0.9969, 0.9870 (0.1, 0.1)	0.9969, 0.987 (1, 10^{-2}, 10^{-2}, 1)	1, 1 ($10^{-5}, 10^{-5}, 10^{-5}$, 0.5)	0.9969, 0.9873 ($10^{-5}, 10^{-5}, 10^{-5}, 10^{-3}$, 4, 0.2)
Average	(0.8557, 0.7315)	(0.8615, 0.7087)	(0.8289, 0.6553)	(0.8471, 0.7540)	(0.8860, 0.8321)
Rank	(3.0000, 2.9167)	(2.6667, 3.1667)	(4.1667, 3.9167)	(2.9167, 2.8333)	(2.2500, 2.1667)
Accuracy(Win,Tie,Loss)	(0, 4, 0)	(1, 2, 1)	(0, 2, 3)	(1, 1, 1)	(3, 1, 0)
F1(Win,Tie,Loss)	(1, 2, 0)	(1, 1, 1)	(0, 2, 2)	(1, 0, 2)	(3, 0, 0)

Table 14
Experiments on NDC datasets*

Dataset	size(train, test)	Accuracy for all models	FTSVM (c_1, c_2, μ) (weight, train)	LSTSVM (c_1, c_2, μ) (train)	LS-ATWSVM ($c_1 = c_3 = c_5, c_2, \mu$) (train)	LSFTSVM-CIL ($c_1 = c_2, c_3 = c_4, c0, \mu$) (weight, train)	Proposed-method (c_1, c_2, c_3, g, μ) (weight, train)
1K	(1000, 10)	100 %	(1, 1, 10^{-5}) (0.018, 1.056)	(1, 1, 10^{-5}) (0.223)	(1, 0.5, 10^{-4}) (0.255)	($10^{-3}, 10^{-5}$, 0.5, 10^{-3}) (0.061, 0.377)	(1, 1, 1, 1, 10^{-4}) (0.992, 0.138)
5K	(5000, 50)	100 %	(1, 1, 10^{-5}) (0.222, 104.615)	(1, 1, 10^{-5}) (40.298)	(1, 0.5, 10^{-4}) (28.519)	($10^{-3}, 10^{-5}$, 0.5, 10^{-3}) (1.806, 10.401)	(1, 1, 1, 1, 10^{-4}) (25.064, 6.179)
10K	(10000, 100)	100 %	(1, 1, 10^{-5}) (0.766, 803.835)	(1, 1, 10^{-5}) (259.786)	(1, 0.5, 10^{-4}) (172.300)	($10^{-3}, 10^{-5}$, 0.5, 10^{-3}) (7.767, 38.204)	(1, 1, 1, 1, 10^{-4}) (208.101, 41.110)
20K	(20000, 200)	100 %	(1, 1, 10^{-5}) (2.627, 6366.038)	(1, 1, 10^{-5}) (1471.069)	(1, 0.5, 10^{-4}) (1016.996)	($10^{-3}, 10^{-5}$, 0.5, 10^{-3}) (31.259, 164.874)	(1, 1, 1, 1, 10^{-4}) (376.230, 457.162)
30K	(30000, 300)	100 %	(1, 1, 10^{-5}) (5.503, 21892.943)	(1, 1, 10^{-5}) (4318.070)	(1, 0.5, 10^{-4}) (3161.030)	($10^{-3}, 10^{-5}$, 0.5, 10^{-3}) (27.379, 315.161)	(1, 1, 1, 1, 10^{-4}) (2292.098, 266.550)
40K	(40000, 400)	100 %	*	*	*	($10^{-3}, 10^{-5}$, 0.5, 10^{-3}) (46.943, 561.312)	(1, 1, 1, 1, 10^{-4}) (4081.265, 302.358)
50K	(50000, 500)	100 %	*	*	*	($10^{-3}, 10^{-5}$, 0.5, 10^{-3}) (75.743, 933.802)	(1, 1, 1, 1, 10^{-4}) (6500.009, 542.674)

*The algorithm execution has failed due to lack of memory.

In terms of scalability, the use of conjugate gradient methods allowed GLSTSVM-CIL to handle large datasets (tested up to size of 50,000 instances). However, the calculation of the gravity weights appeared to be a bottleneck, showing a time complexity with respect to the number of samples. The experimental results (Fig. 7) showed that a large part of the runtime is spent on calculating the weights. It seems that to overcome this limitation, optimization strategies are needed to search for the number of neighborhood points.

Table 15
Medicine and health data

Dataset	FTSVM Accuracy, F1 $(c_1 = c_2, \mu)$ time	LSTSVM Accuracy, F1 $(c_1 = c_2, \mu)$ time	LS-ATWSVM Accuracy, F1 $(c_1 = c_5, c_2, c_3, \mu)$ time	LSFLSTSVM-CIL Accuracy, F1 $(c_0, c_1 = c_2, c_3 = c_4, \mu)$ time	Proposed Method Accuracy, F1 (c_1, c_2, c_3, μ, g) time
Lung cancer	0.9462, 0.5455 (0.1, 0.1) 0.0936	0.9462, 0.5455 (0.1, 0.1) 0.0481	0.957, 0.6000 (0.1, 0.1, 0.1, 0.1) 0.0290	0.9785, 0.8333 $(1, 0.1, 10^{-3}, 10^{-3})$ 0.0180	0.9892, 0.9231 $(10^{-2}, 10^{-5}, 1, 10^{-3}, 5)$ 0.0620
Cervical cancer behavior risk	1, 1 $(10^{-2}, 10^{-2})$ 0.0291	1, 1 $(1, 10^{-2})$ 0.0291	1, 1 (1, 0.1, 0.1, 1) 0.1094	1, 1 $(0.5, 10^{-5}, 10^{-5}, 10^{-5})$ 0.0461	1, 1 $(10^{-5}, 10^{-5}, 10^{-2}, 10^{-3}, 0.8)$ 0.0191
Darwin	0.5283, 0.6914 $(10^{-5}, 10)$ 0.0204	0.5283, 0.6914 $(10^{-5}, 10^{2})$ 0.0048	0.5283, 0.6914 $(10^{-5}, 10^{-4}, 10^{-5}, 10^{-5})$ 0.0098	0.5283, 0.6914 $(0.1, 10^{-5}, 10^{-5}, 10^{-5})$ 0.0040	0.5283, 0.6914 $(1, 10^{3}, 10^{5}, 1, 5)$ 0.0081
Breast cancer coimbra	0.6571, 0.6250 $(1, 10^{-5})$ 0.0191	0.6000, 0.5333 $(1, 10^{-4})$ 0.0046	0.5714, 0.2105 (0.1, 0.1, 0.1, 0.1) 0.0217	0.7429, 0.6667 $(0.5, 10^{5}, 10^{2}, 10^{2})$ 0.0021	0.8286, 0.8235 $(10^{-4}, 10^{-5}, 10^{5}, 10^{-5}, 10)$ 0.0042
Diabetic retinopathy debrecen	0.7312, 0.7304 $(10, 10^{-4})$ 0.4494	0.7225, 0.6863 $(10^{2}, 10^{-4})$ 0.1618	0.6879, 0.6447 $(1, 10^{-5}, 10^{-5}, 1)$ 0.1544	0.6676, 0.7074 $(1.5, 10^{3}, 10, 10)$ 0.2975	0.7052, 0.6600 $(10^{-2}, 10^{-5}, 1, 10^{-5}, 0.2)$ 0.4048
Heart failure clinical records	0.6333, 0.2667 $(10^{-2}, 10^{-4})$ 0.0809	0.6444, 0.2727 $(0.1, 10^{-4})$ 0.0285	0.6111, 0.1026 (0.1, 0.1, 0.1, 0.1) 0.0455	0.6444, 0.2727 $(0.1, 10^{-4}, 10^{-3}, 10^{-3})$ 0.0431	0.6333, 0.2979 $(1, 0.1, 1, 10^{-4}, 3)$ 0.0573
Maternal Health Risk(High)	0.9180, 0.8344 $(10^{2}, 10^{-2})$ 0.3846	0.9180, 0.8299 $(10^{2}, 10^{-2})$ 0.1221	0.9148, 0.8169 $(1, 10^{-2}, 10^{-2}, 1)$ 0.1454	0.8984, 0.7634 $(0.5, 10^{5}, 10^{4}, 10^{4})$ 0.4590	0.9246, 0.8369 $(1062, 10^{4}, 1, 0.1, 10)$ 0.4526
Raisin	0.7148, 0.6638 $(10^{-5}, 10^{-5})$ 0.2364	0.6704, 0.586 $(1, 10^{-5})$ 0.4074	0.6407, 0.5126 (1, 0.1, 0.1, 1) 0.2903	0.6963, 0.6372 $(0.2, 10^{-3}, 10^{-3}, 10^{-3})$ 0.5662	0.7481, 0.7280 $(10^{-3}, 10^{-5}, 10^{-4}, 10^{-5}, 0.4)$ 0.6452
Maternal Health Risk(Low)	0.8361, 0.7664 (10, 1) 0.3671	0.8098, 0.7212 (0.1, 10) 0.1747	0.8066, *underline*0.7704 (1, 0.1, 0.1, 1) 0.1260	0.7475, 0.7004 $(0.5, 10^{-5}, 10^{-2}, 10^{-2})$ 0.4420	0.8164, 0.7053 $(10^{-2}, 10^{-4}, 10^{-3}, 10, 8)$ 0.3804
Parkinsons	0.8644, 0.6364 $(0.1, 10^{-2})$ 0.0691	0.8644, 0.6364 $(1, 10^{-2})$ 0.0245	0.8475, 0.6087 (1, 0.1, 0.1, 1) 0.0299	0.8644, 0.7333 $(2, 10^{-5}, 10^{4}, 10^{4})$ 0.0550	0.8644, 0.6364 $(10^{-3}, 10^{5}, 10^{-4}, 10^{-2}, 0.9)$ 0.0470
Average	(0.7830, 0.6760)	(0.7704, 0.6503)	(0.7565, 0.5958)	(0.7768, 0.7006)	(0.8038, 0.7302)
Rank	(2.60, 2.75)	(3.00, 3.30)	(4.10, 3.90)	(3.20, 2.85)	(2.10, 2.20)
Accuracy(Win,Tie,Loss)	(2, 6, 0)	(0, 6, 0)	(0, 2, 4)	(0, 4, 3)	(4, 4, 0)
F1(Win,Tie,Loss)	(1, 4, 0)	(0, 5, 0)	(1, 2, 5)	(1, 3, 2)	(5, 3, 0)

8. Conclusions

GLSTSVM-CIL introduces a physically inspired approach to class-imbalance learning by incorporating Newtonian Gravitational principles into the support vector machine framework. Its gravitational weighting function is desgined to combine the distance from center and local density. binary-class weighting scaled by imbalance ratio, angular constraints to improve structural risk minimization, and a reformulated quadratic objective solved via conjugate gradients.

Experimental evaluations on synthetic, class imbalance, medical, text, and large-scale datasets demonstrated that GLSTSVM-CIL consistently improves minority-class recognition, The main limitation observed in this method is the increase in the time required to calculate the weights, which we used to obtain by global search of points. In this respect, it works similarly to methods such as FTSVM [6], KNN-STSVM [43] and KNN-LSTSVM [7]. One way to improve performance is to use more optimal methods for finding neighborhoods, which require special machine learning techniques.

On the other hand, calculation of weights is independent of the solution of its optimization problem. Therefore, there is also the potential for parallel processing to improve its performance in terms of time. To address the current challenges, the use of parallel processing techniques, GPU acceleration, and more efficient search methods can be useful. Future work will explore efficient approximations for calculating gravitational weight, incorporate deep learning techniques, develop automatic hyperparameter tuning techniques.

Data availability

Data will be made available on request.

Appendix A. Positive definiteness for Matrix Q

This appendix provides a detailed justification for the positive definiteness of the matrix Q used in our optimization framework, which is crucial for the stable and efficient application of the Conjugate Gradient (CG) method. A symmetric positive definite (SPD) matrix is a prerequisite for the convergence guarantees and numerical stability of CG and related iterative solvers [27,28]. The matrix Q is defined as:

$$Q = \begin{bmatrix} K(B,B) + \frac{c_1}{c_2} S_2^{-2} + E_1 & K(B,A) + E_2 & \phi(B)w_1 + b_1 e_{n_B} \\ K(A,B) + E_2^T & K(A,A) + \frac{c_1}{c_3} S_1^{-2} + E_3 & \phi(A)w_1 + b_1 e_{n_A} \\ w_1^T \phi(B)^T + b_1 e_{n_B}^T & w_1^T \phi(A)^T + b_1 e_{n_A}^T & w_1^T w_1 + b_1^2 \end{bmatrix}$$

where:

- A and B represent sets of data points, with n_A and n_B points, respectively.
- $K(X,Y) = \Phi(X)\Phi(Y)^T$ (or $\Phi(X)^T\Phi(Y)$ depending on Φ convention) is a kernel matrix ($n_X \times n_Y$) derived from a valid kernel function. Valid kernel functions, by definition, generate positive semidefinite (PSD) Gram matrices [30].
- w_1 is a real vector and b_1 is a real scalar, defining a hyperplane.
- e_{n_X} is an $n_X \times 1$ vector of ones.
- E_i are matrices of ones, with dimensions consistent with their respective blocks (E_1 is $n_B \times n_B$, E_2 is $n_B \times n_A$, E_3 is $n_A \times n_A$).
- S_i are diagonal matrices with strictly positive diagonal entries, ensuring S_i^{-2} are also diagonal with strictly positive entries.
- c_i are positive real numbers.

Symmetry of Q

The matrix Q is inherently symmetric. This can be verified by inspection, as $K(X,Y) = K(Y,X)$ for symmetric kernels and the off-diagonal blocks are transposes of each other, ensuring $Q_{ij} = Q_{ji}$.

Proof of Positive Definiteness

A symmetric matrix is positive definite if and only if its quadratic form x^TQx is strictly positive for all non-zero vectors x. Let $x = \begin{pmatrix} x_B \\ x_A \\ \alpha \end{pmatrix}$, where $x_B \in \mathbb{R}^{n_B}$, $x_A \in \mathbb{R}^{n_A}$, and $\alpha \in \mathbb{R}$. The quadratic form is given by:

$$\begin{aligned} x^TQx = {} & x_B^T \left(K(B,B) + \frac{c_1}{c_2} S_2^{-2} + E_1 \right) x_B \\ & + x_A^T \left(K(A,A) + \frac{c_1}{c_3} S_1^{-2} + E_3 \right) x_A \\ & + \alpha^2 (w_1^T w_1 + b_1^2) + 2x_B^T (K(B,A) + E_2) x_A \\ & + 2\alpha x_B^T (\phi(B)w_1 + b_1 e_{n_B}) + 2\alpha x_A^T (\phi(A)w_1 + b_1 e_{n_A}) \end{aligned} \tag{A.1}$$

We establish the strict positivity of x^TQx by analyzing the contribution of each term:

1. Kernel-related terms: The components involving $K(\cdot,\cdot)$ (i.e., $x_B^T K(B,B) x_B + x_A^T K(A,A) x_A + 2x_B^T K(B,A) x_A$) form a quadratic form based on a Gram matrix. Such forms are always non-negative, meaning they contribute a positive semidefinite (PSD) component to x^TQx.

2. Regularization terms from S_i: These are the most critical elements for guaranteeing positive definiteness.

- The term $x_B^T \left(\frac{c_1}{c_2} S_2^{-2} \right) x_B$ is strictly positive for any non-zero x_B. This is because c_1, c_2 are positive scalars and S_2^{-2} is a diagonal matrix with strictly positive entries (as S_2 is defined as a positive diagonal matrix).
- Similarly, $x_A^T \left(\frac{c_1}{c_3} S_1^{-2} \right) x_A$ is strictly positive for any non-zero x_A.

The addition of a symmetric positive definite matrix (even a diagonal one, like $\frac{c_1}{c_2} S_2^{-2}$) to a symmetric positive semidefinite matrix (like $K(B,B)$) results in a symmetric positive definite matrix [28]. These terms effectively act as a strong form of Tikhonov regularization [31], ensuring that the upper-left blocks of Q are well-conditioned and strictly positive definite.

3. Terms from E_i (matrices of ones): These matrices (E_1, E_3 being matrices of all ones) contribute positive semidefinite parts to the quadratic form. For instance, $x_B^T E_1 x_B = (e_{n_B}^T x_B)^2 \geq 0$. The combined E_i terms can be expressed as $(e_{n_B}^T x_B + e_{n_A}^T x_A)^2 \geq 0$, which also contributes a PSD component.

4. Last diagonal term (w_1, b_1): The term $w_1^T w_1 + b_1^2$ represents the squared Euclidean norm of the hyperplane's parameters. Since w_1 and b_1 define a functional hyperplane, they cannot both be zero. Therefore, $w_1^T w_1 + b_1^2 > 0$.

Conclusion on Strict Positivity for Any Non-Zero x

For any non-zero vector $x = \begin{pmatrix} x_B \\ x_A \\ \alpha \end{pmatrix}$:

- If $x_B \neq \mathbf{0}$ or $x_A \neq \mathbf{0}$: The strictly positive contributions from $x_B^T \left(\frac{c_1}{c_2} S_2^{-2} \right) x_B$ and $x_A^T \left(\frac{c_1}{c_3} S_1^{-2} \right) x_A$ (which are guaranteed to be positive for non-zero x_B or x_A respectively) ensure that the overall quadratic form $x^T Q x$ is strictly greater than zero. These dominant positive diagonal contributions overcome any potentially non-positive terms from other components.
- If $x_B = \mathbf{0}$ and $x_A = \mathbf{0}$: Since x is non-zero, α must be non-zero. In this case, the quadratic form simplifies to $x^T Q x = \alpha^2 (w_1^T w_1 + b_1^2)$. As $w_1^T w_1 + b_1^2 > 0$, it follows that $\alpha^2 (w_1^T w_1 + b_1^2) > 0$.

Since $x^T Q x > 0$ for all non-zero vectors x, the matrix Q is definitively positive definite.

References

[1] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (1995) 273–297.
[2] T. Divya, A. Sonali, Twin support vector machine: a review from 2007 to 2014, Egypt. Inf. J. 16 (2015) 55–69.
[3] Jayadeva, R. Khemchandani, C. Suresh, Twin support vector machines for pattern classification, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007) 905–910.
[4] M. Arun Kumar, M. Gopal, Least squares twin support vector machines for pattern classification, Expert Syst. Appl. 36 (2009) 7535–7543.
[5] R. Khemchandani, P. Saigal, S. Chandra, Angle-based twin support vector machine, Ann. Oper. Res. 269 (2018) 387–417.
[6] K. Li, H. Ma, A fuzzy twin support vector machine algorithm, Int. J. Appl. Innov. Eng. Manag. 2 (3) (2013) 459–465.
[7] A. Mir, J.A. Nasiri, KNN-based least squares twin support vector machine for pattern classification, Springer Science + Business Media, LLC, part of Springer Nature 2018 (2018).
[8] I. Hussein, S.A. Anwar, M.I. Ahmad, Imbalanced data classification using SVM based on improved simulated annealing featuring synthetic data generation and reduction, Comput. Mater. Continua 75 (1) (2023) 547–564. https://doi.org/10.32604/cmc.2023.036025
[9] J. Guo, H. Wu, X. Chen, W. Lin, Adaptive SV-borderline SMOTE-SVM algorithm for imbalanced data classification, Appl. Soft Comput. (2023). https://doi.org/10.1016/j.asoc.2023.110986
[10] M.A. Ganaie, M. Tanveer, C.-T. Lin, Large-scale fuzzy least squares twin SVMs for class imbalance learning, IEEE Trans. Fuzzy Syst. 30 (2022) 4815–4827.
[11] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the 17th International Joint Conference on Artificial intelligence (IJCAI), 17, 2001, pp. 973–978.
[12] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[13] J. Tang, et al., A Fast and Robust TSVM for Pattern Classification, arXiv preprint arXiv:1711.05406 (2019).
[14] B. Richhariya, M. Tanveer, A robust fuzzy least squares twin support vector machine for class imbalance learning, Appl. Soft Comput. 71 (2018) 418–432.
[15] M.A. Ganaie, M. Hu, Ensemble learning for imbalanced classification: a review, Artif. Intell. Rev. 54 (6) (2021) 4549–4591.
[16] J.M. Keller, D.J. Hunt, Incorporating fuzzy membership functions into the perceptron algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1985) 693–699.
[17] M.A. Ganaie, M. Tanveer, Fuzzy least squares projection twin support vector machines for class imbalance learning, Appl. Soft Comput. 113 (2021) 107933.
[18] B. Avrim, H. John, K. Ravindran, Foundations of Data Science, 2018.
[19] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.
[20] M. Platt, R. Barr, A gravity-inspired clustering algorithm, Pattern Recognit. Lett. 131 (2020) 283–289.
[21] D. Halliday, R. Resnick, J. Walker, Fundamentals of Physics, A John Wiley & Sons, Inc., Publication, 2021.
[22] A. Mohammadi, J.A. Nasiri, S. Effati, Prediction of chronic diseases with unbalanced data by gravitational support vector machine, in: The 5th National Informatics Conference of Iran, 2024, pp. 47–52.
[23] M.S. Bazaraa, C.M. Shetty, D.S. Hanif, Nonlinear Programming, Theory and Algorithms, A John Wiley & Sons, Inc., Publication, 2006.
[24] F.M. Martin, A scaled conjugate gradient algorithm for fast supervised learning, Neural Netw. 6 (1993) 525–533.
[25] S. Gottlieb, F.F. Paul, Modified conjugate gradient method for the solution of Ax = b, J. Sci. Comput. 13 (2) (1998) 173–183.
[26] J.C. Allwright, Conjugate gradient versus steepest descent, J. Optim. Theory Appl. 20 (1) (1976) 129–134.
[27] J.R. Shewchuk, An Introduction to the Conjugate Gradient Method without the Agonizing Pain (1994). https://www.cs.cmu.edu/~jrs/jrs.html.
[28] G.H. Golub, C.F. Van Loan, Matrix Computations, JHU Press, Baltimore, MD, 4 edition, Baltimore, MD, 2013.
[29] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, New York, NY, 2 edition, New York, NY, 2006.
[30] B. Schölkopf, K. Tsuda, J.-P. Vert, Kernel Methods in Computational Biology, MIT Press, Cambridge, MA, Cambridge, MA, 2004.
[31] A.N. Tikhonov, Solution of incorrectly formulated problems and the regularization method, Dokl. Akad. Nauk SSSR 151 (1963) 501–504.
[32] Alcalà-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. Garcia, L. Sanchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Log. Soft Comput. 17 (2011) 255–287.
[33] D. Dua, C. Graff, UCI Machine Learning Repository, 2017, (http://archive.ics.uci.edu/ml).
[34] D. Cai, "Text Datasets." Zhejiang University, n.d, 2024, (http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html).
[35] C. Sammut, G.I. Webb, TF-IDF, Encyclopedia of Machine Learning, 2010, (https://doi.org/10.1007/978-0-387-30164-8_832).
[36] J. VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media, Inc., 2016.
[37] D.R. Musicant, Normally Distributed Clustered Datasets, 1998. www.cs.wisc.edu/dmi/svm/ndc/.
[38] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[39] R.L. Iman, J.M. Davenport, Approximations of the critical region of the Friedman statistic, Commun. Stat. Theory Methods 9 (6) (1980) 571–595.
[40] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[41] S. García, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2009) 959–977.
[42] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manage. 24 (5) (1988) 513–523.
[43] X. Pan, Y. Luo, Y. Xu, K-nearest neighbor based structural twin support vector machine, Knowl. Based Syst. 89 (2015) 148–158. https://www.sciencedirect.com/science/article/pii/S0950705115003135. https://doi.org/10.1016/j.knosys.2015.07.002

Abdullah Mohammadi is a high school mathematics teacher. In 2022, he began studying machine learning and received his M.Sc. in Data Science from Ferdowsi University of Mashhad in September 2024. Recently, under the supervision of Dr. Sohrab Effati and Dr. Jalal A. Nasiri, he defended his thesis with an excellent grade by presenting a new method in twin support vector machines. His current research focuses on machine learning algorithms for imbalanced data analysis, natural language processing, large language models, and neural network-based learning. He has a particular interest in studying machines' ability to learn cognitively.

Dr. Jalal A. Nasiri is an Assistant Professor at the Department of Computer Science at Ferdowsi University of Mashhad, Iran. From 2016 to 2021, he was Assistant Professor of computational linguistics of Iranian research institute technology for Information Science and Technology (IranDoc). Jalal A. Nasiri obtained his PhD. and M.Sc. in computer engineering from Tarbiat Modares University and Ferdowsi University of Mashhad, Iran, in 2015, and 2009, respectively. His main research interest focuses on Machine Leaning and Natural language Processing (NLP).

Dr. Sohrab Effati received his B.Sc. degree in Applied Mathematics from Birjand University, Birjand, Iran, in 1992, his M.Sc. degree in Applied Mathematics from Kharazmi University, Tehran, Iran, in 1995, and his Ph.D. degree in Control Systems from Ferdowsi University of Mashhad, Mashhad, Iran, in 2000. He began his academic career at Hakim Sabzevari University in 2000 and joined Ferdowsi University of Mashhad in 2008, where he has been a Full Professor in the Department of Applied Mathematics since 2016. His research interests include control theory, optimization, ordinary and partial differential equations, and artificial intelligence, with particular emphasis on their applications in optimization, control problems, and modeling in medical sciences.