A reduced universum twin support vector machine for class imbalance learning

B. Richhariya , M. Tanveer

PII: S0031-3203(19)30451-0
DOI: https://doi.org/10.1016/j.patcog.2019.107150
Reference: PR 107150

To appear in: *Pattern Recognition*

Received date: 22 June 2019
Revised date: 9 October 2019
Accepted date: 4 December 2019

Highlights

- A universum based algorithm is proposed for class imbalance learning.
- Universum learning is used for the first time to solve class imbalance problem.
- Reduced kernel is incorporated for reducing storage and computation cost.
- Proposed approach is useful for large scale class imbalanced datasets.

A reduced universum twin support vector machine for class imbalance learning

B. Richhariya[1], M. Tanveer[2]

Discipline of Mathematics, Indian Institute of Technology Indore, Simrol, Indore, 453552, India

Abstract

In most of the real world datasets, there is an imbalance in the number of samples belonging to different classes. Various pattern classification problems such as fault or disease detection involve class imbalanced data. The support vector machine (SVM) classifier becomes biased towards the majority class due to class imbalance. Moreover, in the existing SVM based techniques for class imbalance, there is no information about the distribution of data. Motivated by the idea of prior information about data distribution, a reduced universum twin support vector machine for class imbalance learning (RUTSVM-CIL) is proposed in this paper. For the first time, universum learning is incorporated with SVM to solve the problem of class imbalance. Oversampling and undersampling of data is performed to remove the imbalance in the classes. The universum data points are used to give prior information about the data. To reduce the computation time of our universum based algorithm, we use a small sized rectangular kernel matrix. The reduced kernel matrix needs less storage space, and thus applicable for large scale imbalanced datasets. Comprehensive experimentation is performed on various synthetic, real world and large scale imbalanced datasets. In comparison to the existing approaches for class imbalance, the proposed RUTSVM-CIL gives better generalization performance for most of the benchmark datasets. Also, the computation cost of RUTSVM-CIL is very less, making it suitable for real world applications.

Keywords: Universum, rectangular kernel, class imbalance, imbalance ratio, twin support vector machine.

[1] phd1701241001@iiti.ac.in

[2] Corresponding author
E-mail address: tanveergouri@gmail.com, mtanveer@iiti.ac.in

1. Introduction

Support vector machine (SVM) [1] is one of the most widely used algorithms for classification problems such as disease detection [2, 3], and fault diagnosis [4]. It gives a unique globally optimal solution for the optimization problem. This is in contrast to the local minimal solution of artificial neural network (ANN). SVM also incorporates the structural risk minimization (SRM) principle by using a maximum margin approach. Various variants of SVM have been proposed to reduce its computational cost while improving its generalization ability. In order to improve the efficiency of SVM, a twin support vector machine (TWSVM) is proposed [5], where two non-parallel hyperplanes are obtained by solving quadratic programming problems (QPPs). In TWSVM, the classifying hyperplanes tends to remain far away from the other class and closer to their own class. Kumar and Gopal [6] reduced the computation time of TWSVM by proposing a least squares twin support vector machine (LSTSVM), which involves the solution of a pair of system of linear equations. To improve the sparsity of TWSVM, a $\nu-$twin support vector machine ($\nu-$TSVM) is also proposed [7]. For adding robustness to the TWSVM model, L1-norm based twin support vector machines are proposed [8, 9]. A $L_{2,1}$ -norm based enhanced multi-weight vector projection support vector machine ($L_{2,1}$-EMVSVM) is proposed [10] to deal with outliers. Structural information is incorporated in TWSVM using a weighting based technique [11] to remove the effect of noise. In order to improve the generalization performance, multi-view learning is also used with non parallel hyperplane based SVM models [12, 13]. An efficient best fitting hyperplane method is proposed [14] to improve the sparsity of non parallel hyperlane based models. For multiclass classification, a weighted linear loss multiple birth support vector machine is proposed [15] based on information granulation.

Weston et al. [16] came up with the idea of universum support vector machine (USVM) to incorporate prior information about distribution of data in the construction of the classifier. Chapelle et al. [17] stated that the selection of universum data is dependent on the type of problem. To reduce the training time of USVM, a universum twin support vector machine (UTSVM) is proposed in [18]. An improvement on training time of UTSVM is proposed as universum least squares twin support vector machine with universum data (ULSTSVM) [19]. In most of the cases, the random averaging of data points [2, 18, 19] is used for the generation of universum data. However, the existing universum based algorithms are expensive with regard to computation cost. The addition of universum data points in the optimization problem of SVM leads to more training time [2].

In the class imbalance scenario, the SVM classifier gets biased towards the majority class, which is having more number of data points. Consequently, the data points of the minority class get misclassified. In applications such as disease diagnosis [2, 3], more emphasis is on the correct classification of the minority class. Various researchers have proposed different techniques for SVM to deal with this problem. By introducing fuzzy membership values for the data points, a fuzzy support vector machine (FSVM-CIL) is proposed for class imbalance [20]. A boosting algorithm termed as boosting support vector machine (BSVM) is proposed [21] for creating a balance in the classes. Chawla et al. [22] proposed an oversampling technique termed as SMOTE (Synthetic minority oversampling technique) to overcome the imbalance problem. Tang et al. [23] discussed the models of SVM for highly imbalanced data, while undersampling approach for class imbalance data is discussed in [24, 25].

Various other approaches for dealing with class imbalance in SVM are proposed such as fuzzy weighting based techniques [26]. An entropy based fuzzy support vector machine (EFSVM) [27] is proposed for class imbalance using information entropy of data. A robust fuzzy least squares twin support vector machine (RFLSTSVM-CIL) [28] is proposed with a novel fuzzy membership function for class imbalanced data. However, the drawback with all the fuzzy based approaches is the additional computational time for calculation of the fuzzy memberships of data points. A novel general twin support vector with pinball loss (Pin-GTSVM) [29] is proposed for class imbalance with noisy data. An alternative approach for imbalanced data is proposed as maximum margin of twin spheres support vector machine (MMTSSVM) [30] for class imbalanced data, where hyperspheres are obtained instead of hyperplanes. To circumvent the problem of noise, pinball loss function is utilized to propose a maximum margin and minimum volume hyper-spheres machine (Pin-M^{3}HM) [31] for imbalanced data classification. In order to create balance between the classes, a k-nearest neighbour (KNN) based strategy is also used with hypersphere based models in [32].

Lee and Mangasarian [33] proposed a reduced support vector machines (RSVM) to restrict the number of support vectors, and reduce the kernel matrix by randomly selecting a subset of training data considered as candidates of support vectors. For regression problems, Singh et al. [34] proposed a reduced twin support vector regression (RTSVR). A statistical analysis of RSVM is presented in [35] emphasizing on the fact that the reduced kernel retains most of the information for learning.

Universum based algorithms give better generalization performance [16, 17] due to inclusion of universum data in the optimization problem. However, the model becomes computationally expensive [2, 19]. In order to apply the universum learning for class imbalance data, we propose an efficient algorithm termed as reduced universum twin support vector machine for class imbalance learning (RUTSVM-CIL).

The main contributions of this work are as follows:

- To include prior information about data distribution in the classification of imbalanced data, the idea of universum is incorporated in the proposed RUTSVM-CIL. As per our survey, the concept of universum is used for the first time to solve class imbalance problem.

- Universum based learning algorithms have drawback of higher training time. So, we propose an efficient RUTSVM-CIL algorithm to enable universum based learning for application on large datasets. Various experiments are performed on large scale imbalanced datasets to justify the applicability of the proposed RUTSVM-CIL.
- Motivated by the widely used undersampling approach for imbalanced data [23-25], we applied the concept of undersampling in the constraints as well as in the kernel matrix of UTSVM. This leads to equal weights to the two classes in the construction of the classifier. The reduced kernel matrix is constructed from a randomly selected subset of the original kernel matrix, which leads to less computation complexity and less storage requirement for the kernel matrices.
- The proposed reduced universum twin support vector machine (RUTSVM-CIL) use the universum samples for balancing the imbalanced data. This gives prior information about the distribution of data with lesser computation cost in comparison to existing algorithms.
- The concept of reduced kernel is not utilized with universum based support vector machine algorithms in the past. The proposed model includes this idea, and gives better classification accuracy with less computation time for class imbalanced datasets.

In this work, we assume all vectors as column vectors in $n-$dimensional real space R^n. $||v||$ is the notation for 2-norm of a vector, and $||M||$ represents the 2-norm of a matrix M. The inner product of two vectors is denoted as $v^t w$ where v^t is the transpose of v. The symbol I denotes the identity matrix, and e denotes the vector of ones of appropriate dimension. The label for the minority and majority class are assigned as '+1' and '-1' respectively. Therefore, in this paper we use the terms for minority and majority class as positive and negative class respectively. The formula for the imbalance ratio (IR) is

$$\text{IR} = \frac{\text{Number of majority class samples}}{\text{Number of minority class samples}}.$$

This paper is organized as follows: section 2 discusses the formulation of UTSVM in brief. Section 3 presents the formulation of the proposed method RUTSVM-CIL. The analysis of the proposed algorithm is shown in section 4. Numerical experiments on several synthetic, real world, and large scale datasets are presented in section 5. Section 6 gives the conclusions with future directions.

2. Related work

In this section, we briefly discuss the formulation of universum twin support vector machine (UTSVM).

2.1 Universum twin support vector machine (UTSVM)

In 2012, Qi et al. [18] proposed a novel algorithm termed as universum twin support vector machine (UTSVM) to give prior information about the distribution of data. The formulation of UTSVM is described in the following:

Consider the matrices X_1 and X_2 representing the data points of 'class 1' and 'class 2' of size $r \times n$ and $s \times n$ respectively, where r and s are total number of data points of 'class 1' and 'class 2' respectively. The number of features is denoted by n. Here, the matrix U contains the universum data points of size $u \times n$. In UTSVM, the universum data points satisfy the constraint of lying in between an $\varepsilon -$ insensitive tube.

The optimization problems of linear UTSVM [18] with the universum constraints are written as

$$\begin{aligned} \min_{w_1, b_1, \xi, \psi} \quad & \frac{1}{2} \| \mathrm{X}_1 w_1 + e_1 b_1 \|^2 + C_1 e_2^t \xi + C_u e_u^t \psi \\ \text{s. t.} \quad & -(\mathrm{X}_2 w_1 + e_2 b_1) + \xi \geq e_2, \ \xi \geq 0 \\ & (\mathrm{U} w_1 + e_u b_1) + \psi \geq (-1+\varepsilon) e_u, \ \psi \geq 0 \end{aligned} \tag{1}$$

$$\begin{aligned} \min_{w_2, b_2, \eta, \psi^*} \quad & \frac{1}{2} \| \mathrm{X}_2 w_2 + e_2 b_2 \|^2 + C_2 e_1^t \eta + C_u e_u^t \psi^* \\ \text{s. t.} \quad & (\mathrm{X}_1 w_2 + e_1 b_2) + \eta \geq e_1, \ \eta \geq 0 \\ & -(\mathrm{U} w_2 + e_u b_2) + \psi^* \geq (-1+\varepsilon) e_u, \ \psi^* \geq 0 \end{aligned} \tag{2}$$

where ξ, η, ψ, ψ^* represent slack variables, C_1, C_2, C_u are penalty parameters, ε is the tolerance of the ε - insensitive tube, and e_1, e_2, e_u are vectors of ones of appropriate dimensions.

The Wolfe duals of QPPs in (1) and (2) can be obtained by applying the Karush-Kuhn-Tucker (K.K.T.) conditions as

$$\begin{aligned} \max_{\alpha_1, \mu_1} \quad & e_2^t \alpha_1 - \frac{1}{2} (\alpha_1^t T - \mu_1^t O)(S^t S)^{-1} (T^t \alpha_1 - O^t \mu_1) + (\varepsilon - 1) e_u{}^t \mu_1 \\ \text{s. t.} \quad & 0 \leq \alpha_1 \leq C_1, \ \ 0 \leq \mu_1 \leq C_u \end{aligned} \tag{3}$$

$$\begin{aligned} \max_{\alpha_2, \mu_2} \quad & e_1^t \alpha_2 - \frac{1}{2} (\alpha_2^t S - \mu_2^t O)(T^t T)^{-1} (S^t \alpha_2 - O^t \mu_2) + (\varepsilon - 1) e_u{}^t \mu_2 \\ \text{s. t.} \quad & 0 \leq \alpha_2 \leq C_2, \ \ 0 \leq \mu_2 \leq C_u \end{aligned} \tag{4}$$

where $S = [\mathrm{X}_1 \ e_1]$, $T = [\mathrm{X}_2 \ e_2]$, $O = [\mathrm{U} \ e_u]$, and $\alpha_1, \alpha_2, \mu_1, \mu_2$ are the vectors of Lagrange multipliers.

We compute the hyperplanes $w_1^t x + b_1 = 0$ and $w_2{}^t x + b_2 = 0$ by computing the values of w_1, w_2, b_1 and b_2 using the following equations (5) and (6),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(S^t S + \sigma I)^{-1} (T^t \alpha_1 - O^t \mu_1) \,, \tag{5}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (T^t T + \sigma I)^{-1} (S^t \alpha_2 - O^t \mu_2) \,, \tag{6}$$

where σ is a small positive value to deal with the case of singular matrices.

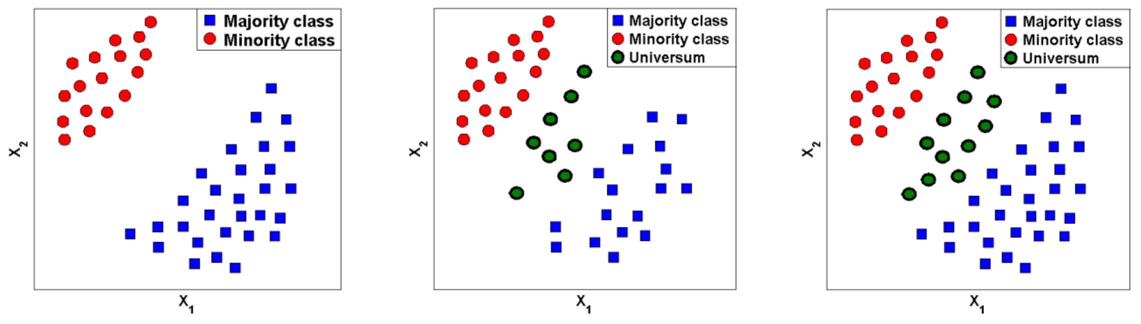
A new data point $x \in R^n$ is classified to a given class $'i'$ by using the following formula:

$$class\ i = \min |w_i^t\ x + b_i| \quad \text{for } i = 1,2. \tag{7}$$

3. Proposed reduced universum twin support vector machine for class imbalance learning (RUTSVM-CIL)

The formulation of UTSVM [18] involves the solution of two small QPPs as compared to USVM [16] where one large QPP is solved. In universum based SVM algorithms, the addition of the universum data points increases the computational complexity of the algorithm [2, 16, 18]. Also, universum based algorithms like USVM and UTSVM suffer from the problem of class imbalance. To remove these drawbacks and to give prior information about data, we propose a reduced universum twin support vector machine for class imbalance learning (RUTSVM-CIL). We utilize the concepts of undersampling and oversampling to formulate an algorithm specifically for class imbalance. Moreover, we incorporate information about data distribution in the formulation from majority class data points. Further, the training time is reduced by constructing the rectangular kernel matrix from the undersampled dataset in proposed RUTSVM-CIL.

In RUTSVM-CIL, the data points of majority (negative) class are reduced by using random undersampling approach [23-25] as shown in Fig. 1(b). This leads to a balance condition for construction of the hyperplanes as well as reduces the training time. The reduced kernel also utilizes the widely used undersampling approach for imbalanced data in the kernel matrix. In the proposed RUTSVM-CIL, the universum data points are selected using the random averaging scheme [2, 18].



(a) Imbalanced data (b) Case 1 (c) Case 2

Figure 1: Balancing of class imbalanced data with universum in proposed RUTSVM-CIL. The distributions of data points in the optimization problem of minority class and majority class hyperplanes in proposed RUTSVM-CIL are shown in (b) and (c) respectively.

3.1 Universum for class imbalance

In the case of class imbalance, most of the information about data distribution is contained in the majority class. So, we increase the number of universum data points in the case of the majority class hyperplane to give prior information of the data. In Fig. 1(c), we treat universum data points as belonging to minority class. This is in contrast to UTSVM where universum is treated as not belonging to any of the binary classes, and do not solve the class imbalance problem. We add additional constraints of universum points having size equal to difference in the number of data points of the two classes. This gives constraints in the construction of the majority class hyperplane to keep universum data points at a distance of $(1-\varepsilon)$ from the majority class. It also prevents the majority class hyperplane to lie closer to the minority class data points, while giving prior information about the data. Thus, the biasing of the classifier towards the majority class is reduced, and the generalization performance is improved using this scheme.

In case of minority class i.e., positive class, random undersampling of the negative class is used to create the balance situation for the construction of the positive class hyperplane. The same reduced kernel matrix is used in the construction of both the hyperplanes. The formulation of proposed RUTSVM-CIL is as follows:

Consider the matrices X_1 and X_2 representing the data points of 'class 1' and 'class 2', having dimension $r \times n$ and $s \times n$ respectively. X_2^* is the randomly chosen reduced data samples of the negative class of size $r \times n$. The universum data points are contained in the matrix U of size $d \times n$, where d is the difference in the number of data points of the two classes i.e., $(s-r)$. The matrix U^* is a subset of U of size equal to ceiling of $\frac{r}{2}$ denoted as g, and the dimension of each data point is represented by n. The data points in U^* are selected randomly from the set U.

3.2 Linear RUTSVM-CIL

The optimization problems of linear RUTSVM-CIL in primal are written as

$$\min_{w_1, b_1, \xi, \psi} \ \frac{1}{2}\| X_1 w_1 + e_1 b_1 \|^2 + C_1 e_2^t \xi + C_u e_g^t \psi$$

$$\text{s. t.} \quad -(X_2^{*} w_1 + e_1 b_1) + \xi \geq e_1, \ \xi \geq 0 \tag{8}$$

$$(U^{*} w_1 + e_g b_1) + \psi \geq (-1+\varepsilon) e_g \, , \ \psi \geq 0$$

$$\min_{w_2, b_2, \eta, \psi^*} \ \frac{1}{2}\| X_2 w_2 + e_2 b_2 \|^2 + C_2 e_1^t \eta + C_u e_d^t \psi^*$$

$$\text{s. t.} \quad (X_1 w_2 + e_1 b_2) + \eta \geq e_1, \ \eta \geq 0 \tag{9}$$

$$(U w_2 + e_d b_2) + \psi^* \geq (1-\varepsilon) e_d \, , \ \psi^* \geq 0 \, ,$$

where ξ, ψ, η, ψ^* represent the slack variables; C_1, C_2 and C_u represent the penalty parameters; e_1, e_2, e_g and e_d are vectors of ones of appropriate dimensions.

Here, QPP (8) corresponds to the optimization problem for the positive class, and QPP (9) corresponds to the negative class hyperplane. One can observe in (8) that the constraint on the universum is to lie within a distance of $(1-\varepsilon)$ from the positive class hyperplane, whereas in (9) the constraint on universum is to lie away from the negative class hyperplane by a distance of $(1-\varepsilon)$. So, in QPP (8), the universum provides the prior information about the data, and in QPP (9) the prior information is provided in a manner that removes the biasing of the classifier.

The Wolfe duals of the primal problems (8) and (9) are obtained by applying the K.K.T. conditions as

$$\max_{\alpha_1, \mu_1} \ e_1^t \alpha_1 - \frac{1}{2}(\alpha_1^t T^* - \mu_1^t O^*)(S^t S)^{-1}(T^{*t}\alpha_1 - O^{*t}\mu_1) + (\varepsilon - 1) e_g{}^t \mu_1 \tag{10}$$

$$\text{s. t.} \quad 0 \leq \alpha_1 \leq C_1, \ 0 \leq \mu_1 \leq C_u \, ,$$

$$\max_{\alpha_2, \mu_2} \ e_1^t \alpha_2 - \frac{1}{2}(\alpha_2^t S + \mu_2^t O)(T^t T)^{-1}(S^t \alpha_2 + O^t \mu_2) + (1-\varepsilon) e_d{}^t \mu_2 \tag{11}$$

$$\text{s. t.} \quad 0 \leq \alpha_2 \leq C_2, \ 0 \leq \mu_2 \leq C_u \, ,$$

where $S = [X_1 \ \ e_1]$, $T^* = [X_2^{*} \ \ e_1]$, $T = [X_2 \ \ e_2]$, $O^* = [U^* \ \ e_g]$, $O = [U \ \ e_d]$, and $\alpha_1, \alpha_2, \mu_1, \mu_2$ are the vectors of Lagrange multipliers.

The hyperplanes $x^t w_1 + b_1 = 0$ and $x^t w_2 + b_2 = 0$ are obtained using the value of the parameters w and b from the following equations (12) and (13),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(S^t S + \sigma I)^{-1}(T^{*t}\alpha_1 - O^{*t}\mu_1), \tag{12}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (T^t T + \sigma I)^{-1}(S^t\alpha_2 + O^t\mu_2), \tag{13}$$

where σ is a small positive value to deal with case when $S^t S$ and $T^t T$ are singular matrices. For the classification of a new data point $x \in R^n$, the following decision function is used

$$class\ i = \min |x^t w_i + b_i| \text{ for } i = 1,2. \tag{14}$$

3.3 Non-linear RUTSVM-CIL

The optimization problems of non-linear RUTSVM-CIL are expressed using the kernel matrices as follows:

$$\begin{aligned} \min_{w_1, b_1, \xi, \psi} \quad & \frac{1}{2}\| K(X_1, D^t)w_1 + e_1 b_1 \|^2 + C_1 e_2^t \xi + C_u e_g^t \psi \\ \text{s. t.} \quad & -(K(X_2^*, D^t)w_1 + e_1 b_1) + \xi \geq e_1,\ \xi \geq 0 \\ & (K(U^*, D^t)w_1 + e_g b_1) + \psi \geq (-1+\varepsilon)e_g,\ \psi \geq 0 \end{aligned} \tag{15}$$

$$\begin{aligned} \min_{w_2, b_2, \eta, \psi^*} \quad & \frac{1}{2}\| K(X_2, D^t)w_2 + e_2 b_2 \|^2 + C_2 e_1^t \eta + C_u e_d^t \psi^* \\ \text{s. t.} \quad & (K(X_1, D^t)w_2 + e_1 b_2) + \eta \geq e_1,\ \eta \geq 0 \\ & (K(U, D^t)w_2 + e_d b_2) + \psi^* \geq (1-\varepsilon)e_d,\ \psi^* \geq 0, \end{aligned} \tag{16}$$

where the slack variables are ξ, ψ, η, ψ^*; the penalty parameters are represented by C_1, C_2 and C_u; $D = [X_1; X_2^*]$ is the reduced kernel; e_1, e_2, e_g and e_d are vectors of ones of appropriate dimension, and $K(x^t, D^t) = (k(x, x_1), k(x, x_2) \ldots, k(x, x_{(2r)}))$ is a row vector of the reduced kernel matrix in R^{2r} space, where r represents the number of data points belonging to 'class 1'.

The Wolfe duals of primal problems in (15) and (16) are obtained using the K.K.T. conditions as

$$\max_{\alpha_1,\mu_1} \; e_1^t\alpha_1 - \frac{1}{2}(\alpha_1^t F^* - \mu_1^t P^*)(E^t E)^{-1}(F^{*t}\alpha_1 - P^{*t}\mu_1) + (\varepsilon - 1)e_g{}^t\mu_1 \tag{17}$$

s. t. $0 \le \alpha_1 \le C_1,\ 0 \le \mu_1 \le C_u,$

$$\max_{\alpha_2,\mu_2} \; e_1^t\alpha_2 - \frac{1}{2}(\alpha_2^t E + \mu_2^t P)(F^t F)^{-1}(E^t\alpha_2 + P^t\mu_2) + (1-\varepsilon)e_d{}^t\mu_2 \tag{18}$$

s. t. $0 \le \alpha_2 \le C_2,\ 0 \le \mu_2 \le C_u,$

where $E = [K(\mathrm{X}_1, D^t)\ \ e_1]$, $F^* = [K(\mathrm{X}_2{}^*, D^t)\ \ e_1]$, $F = [K(\mathrm{X}_2, D^t)\ \ e_2]$, $P^* = [K(\mathrm{U}^*, D^t)\ \ e_g]$ and $P = [K(\mathrm{U}, D^t)\ \ e_d]$.

The non-linear hyperplanes $K(x^t, D^t)w_1 + b_1 = 0$ and $K(x^t, D^t)w_2 + b_2 = 0$ are obtained by using parameters w and b from the following equations (19) and (20),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(E^t E + \sigma I)^{-1}(F^{*t}\alpha_1 - P^{*t}\mu_1), \tag{19}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (F^t F + \sigma I)^{-1}(E^t\alpha_2 + P^t\mu_2), \tag{20}$$

where σ is a small positive value to deal with the case of singular matrices.

Every new data point $x \in R^n$ is classified to a class $'i'$ using the following formula:

$$class\ \ i = \min |K(x^t, D^t)w_i + b_i| \quad \text{for } i = 1,2. \tag{21}$$

The algorithm for RUTSVM-CIL is briefly described in Alg. 1.

Algorithm 1: RUTSVM-CIL.

Input: $\{X_1\}_{r\times n}$, $\{X_2\}_{s\times n}$, $\{U\}_{d\times n}$, $d = s - r$ and $g = ceil\ (r/2)$.

Output: The weight vectors and bias i.e, w_i, b_i, $i = 1, 2$ for ‘class 1’ and ‘class 2’ respectively.

1. Construct matrices $\{X_2{}^*\}_{r\times n}$ and $\{U^*\}_{g\times n}$ using randomly selected data samples of negative class i.e., $\{X_2\}_{s\times n}$ and universum $\{U\}_{d\times n}$ respectively.
2. Set the constraints of optimization problem of positive class hyperplane using matrices $\{X_2{}^*\}_{r\times n}$ and $\{U^*\}_{g\times n}$, and for negative class using matrix $\{X_1\}_{r\times n}$ with universum $\{U\}_{d\times n}$ treated as belonging to positive class.
3. Solve the QPPs in the dual form to obtain the Lagrange multipliers $\{\alpha_{1i}\}_{i=1}^{r}$, $\{\mu_{1i}\}_{i=1}^{g}$ and $\{\alpha_{2i}\}_{i=1}^{r}$, $\{\mu_{2i}\}_{i=1}^{d}$.
4. Calculate w_i, b_i, $i = 1, 2$ using the Lagrange multipliers obtained in step 3.
5. Return w_i, b_i, $i = 1, 2$ for the construction of hyperplanes of the two classes.

4. Analysis of proposed algorithm

In case of class imbalance problems, knowledge about the distribution of data is contained in the majority class, simply due to the more number of data points in it. To incorporate this knowledge of the distribution as prior information in the construction of the classifier, the proposed model uses the concept of universum. Moreover, the traditional approaches does not resolve the problem of computation time for the class imbalance problems, and in some cases the use of fuzzy membership functions incur additional computation time in the training phase. For reducing this computational cost, the proposed RUTSVM-CIL uses the concept of a rectangular kernel of smaller size which reduces the training time of the algorithm significantly. By using the reduced kernel, the setup time [34] for solving the QPP is reduced. The setup time consists of the construction of kernel matrices and computation of the inverses. In comparison to UTSVM, the setup time as well as time for solving the QPP is reduced by using the rectangular kernel.

The proposed RUTSVM-CIL utilizes the benefits of undersampling as well as oversampling, with reduced kernel for UTSVM to classify imbalanced datasets. There is undersampling of the majority class as stated in subsection 4.1, and oversampling of universum samples in 4.2, to create a balance situation for classification. The proposed scheme for the construction of twin hyperplanes in our RUTSVM-CIL is discussed below:

4.1 Positive (minority) class hyperplane

The negative class has more number of data points as compared to the positive class. To reduce the biasing of the positive class hyperplane, a random undersampling approach [23-25] is used to reduce the number of data points of the negative class. Here, the universum data points are also used by randomly selecting data points from the set U. This creates a balance situation for the construction of positive hyperplane. Moreover, the computation time for calculating the inverse of the matrix is reduced due to the undersampling. Experimental analysis on selection of universum is presented in section 5.3.5.

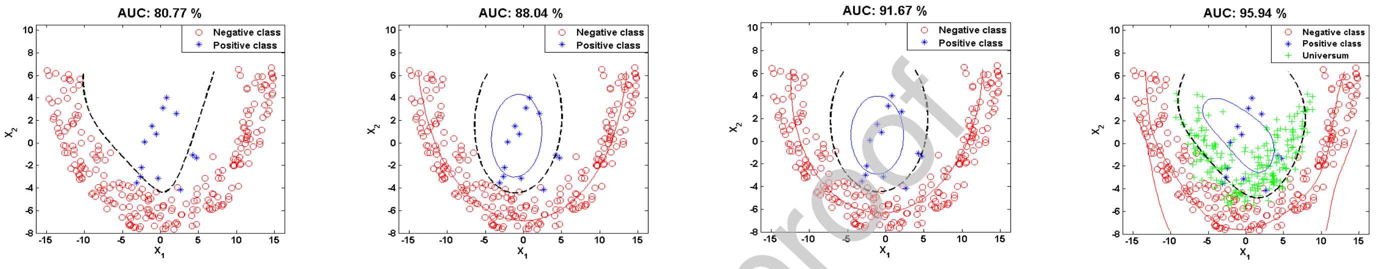
4.2 Negative (majority) class hyperplane

In case of negative class hyperplane, the random undersampling of the negative samples would result into a poor approximation of the actual data. So, we used all the samples of both classes. Now, to create a balance situation for the negative hyperplane, we used more number of universum data points U in the constraints which reduce the bias towards the negative class. This also gives prior information in the construction of the classifier.

The universum data points are kept at a distance of $(1-\varepsilon)$ from the classifier in QPP (9), and the minority class is unit distance away from the negative class hyperplane. So, the universum points help in providing the balance between the two classes. Thus, the majority class hyperplane gets some information about the distribution of data, as well as do not get biased towards its own class. It is clearly visible in Figs. 2 and 3 that the classifier in the proposed RUTSVM-CIL aligns itself with the distribution of data. Moreover, the classifier is also not biased towards the majority class, leading to better classification of samples in each class.

4.3 Kernel matrix

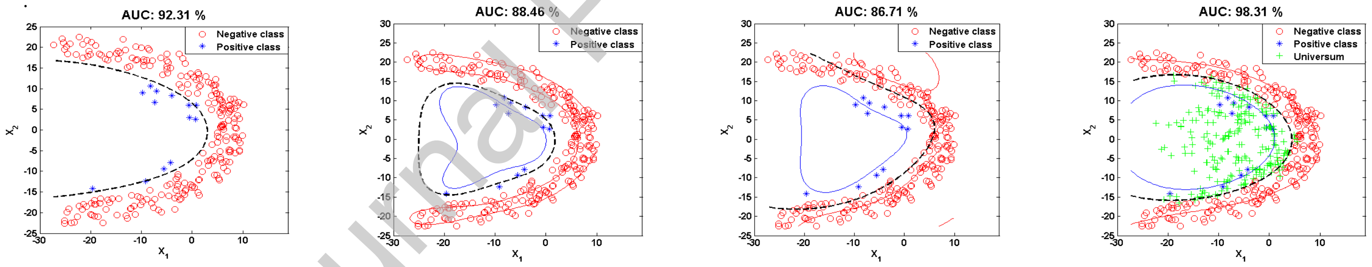
To incorporate the concept of undersampling in the kernel matrix, we choose the rectangular kernel, where the size of the kernel depends on the size of the minority class. The columns of the kernel matrix are equal to twice the size of the minority class, which includes the positive class of size r, and randomly selected samples of size r from the negative class. This also creates a balance in the kernel matrix w.r.t. kernel mapping of the data. So, if there is more imbalance in the data, then lesser



EFSVM TWSVM TWSVM-RUS RUTSVM-CIL

Figure 2: Performance of EFSVM, TWSVM, TWSVM-RUS and proposed RUTSVM-CIL for the classification of synthetic dataset Crescent_&_full_moon using Gaussian kernel function.

.



EFSVM TWSVM TWSVM-RUS RUTSVM-CIL

Figure 3: Performance of EFSVM, TWSVM, TWSVM-RUS and proposed RUTSVM-CIL for the classification of synthetic dataset Half_kernel using Gaussian kernel.

will be the size of the kernel, and lesser the computation time. The size of universum data for the majority class is directly proportional to the imbalance ratio (IR) of data, while the size of the kernel matrix is inversely proportional to the imbalance ratio. However, the training time of the proposed RUTSVM-CIL is inversely proportional to the imbalance ratio of data, since the construction of kernel matrices and calculating the inverses are computationally expensive steps. Experiments on IR vs. time are presented in section 5.3.4. The proposed RUTSVM-CIL also requires less memory to store the kernel matrices, and is suitable for handling large scale imbalanced datasets.

4.4 Computational complexity

Let number of samples belonging to positive class be m_1 and number of samples belonging to negative class be m_2, then $m = m_1 + m_2$, and $IR = m_2 / m_1$ where IR is the imbalance ratio. In the case of TWSVM, the time complexity is given as follows [5]:

$$T = O(m_2)^3 + O(m_1)^3,$$

$$\Rightarrow\ T = O(IR * m_1)^3 + O(m_1)^3,$$

$$\Rightarrow\ T = (IR^3 + 1)\, O(m_1)^3. \tag{22}$$

If the imbalance ratio i.e, IR is equal to 1, then $T = 2 * O\left(\frac{m}{2}\right)^3$ as stated in [5].

In the proposed RUTSVM-CIL, the computational complexity is given by considering the constraints in both QPPs (8 & 9) as follows:

$$T = O\left(m_1 + \frac{m_1}{2}\right)^3 + O(m_1 + (m_2 - m_1))^3,$$

$$\Rightarrow\ T = O\left(m_1 + \frac{m_1}{2}\right)^3 + O(m_2)^3,$$

$$\Rightarrow\ T = (IR^3 + 3.375)\, O(m_1)^3. \tag{23}$$

The time complexity of the proposed RUTSVM-CIL is comparable to TWSVM (Eq. 22). However, since we include the reduced kernel in constructing the kernel matrices in the proposed approach, the setup time involving the construction of kernel matrices and computation of inverses is reduced. This leads to less computation complexity of RUTSVM-CIL as compared to TWSVM.

In TWSVM, two kernel matrices are computed i.e, for the positive and negative class. The computation complexity of calculating the kernel matrices in TWSVM is given as follows:

$$T = O(m_1 * m) + O(m_2 * m)$$

$$\Rightarrow\ T = O(m_1 * (m_1 + IR * m_1)) + O(IR * m_1 * (m_1 + IR * m_1)),$$

$$\Rightarrow\; T=(1+2\,IR+IR^2)O(m_1{}^2). \tag{24}$$

In proposed RUTSVM-CIL, three kernel matrices are computed i.e., for positive and negative class, and the universum. The computation complexity of calculating the kernel matrices for RUTSVM-CIL is given as follows:

$$T=O(m_1*2\,m_1)+O(m_2*2\,m_1)+O((m_2-m_1)*2\,m_1)$$

$$\Rightarrow\; T=O(2\,m_1{}^2)+O(IR*m_1*2\,m_1)+O((IR*m_1-m_1)*2\,m_1),$$

$$\Rightarrow\; T=(1+IR+IR-1)*2\,O(m_1{}^2),$$

$$\Rightarrow\; T=(4\,IR)\,O(m_1{}^2). \tag{25}$$

It is evident from Eqs. (24) and (25) that in comparison to TWSVM, proposed RUTSVM-CIL is computationally efficient for datasets with high imbalance ratios.

Further, the time complexity in calculating inverse of matrix having size $m\times m$ is $O(m)^3$. So, for TWSVM the complexity is $2O(m_1)^3(1+IR)^3$, while the proposed RUTSVM-CIL has very less complexity i.e., $2O(2m_1)^3$. Therefore, the complexity of RUTSVM-CIL is less than TWSVM for $IR>1$. This makes our RUTSVM-CIL suitable for large scale imbalanced datasets.

5. Numerical experiments

In this section, experiments are performed on various synthetic as well as real world benchmark datasets for the comparison of the proposed RUTSVM-CIL with existing approaches. The proposed method is compared with EFSVM [27], SVM-RUS [23-25], TWSVM [5], TWSVM-RUS [23-25], TWSVM-SMOTE [22], MMTSSVM [30], FTSVM [7, 26] and UTSVM [18] in terms of classification accuracy and training time. We also performed experiments on large scale imbalanced datasets to justify the applicability of the proposed approach.

5.1 Experimental setup

The experiments for all the algorithms are performed on a computer having Windows 10 OS, 64 bit, with 3.60 GHz Intel® core^{TM} i7-7700 processor and 16 GB RAM. The algorithms are implemented in MATLAB R2008b environment. To solve the QPPs, MOSEK optimization toolbox (http://www.mosek.com) is utilized. Experiments on large datasets are carried out on a workstation with 64-bit Windows 10 OS, running on 2.30 GHz Intel® Xeon processor, and 128 GB RAM. For the kernel matrices, Gaussian kernel is used for the kernel function in all the algorithms, defined as $K(b,c)=\exp\left(-\frac{1}{2\mu^2}\|b-c\|^2\right)$ where vector $b,c\in R^n$, and μ is kernel parameter. The results for all the algorithms are calculated for the non-linear case using Gaussian kernel. For comparison of classification accuracy of the methods, area under receiver operating characteristics (ROC) curve i.e. AUC [27] is used which is defined as

$$\mathrm{AUC} = \frac{1 + \mathrm{TP}_{\mathrm{rate}} - \mathrm{FP}_{\mathrm{rate}}}{2},$$

where $\mathrm{TP}_{\mathrm{rate}}$ is the true positive classification of positive class, and $\mathrm{FP}_{\mathrm{rate}}$ is the false positive classification of negative class data points.

The penalty parameters $C = C_1 = C_2 = C_u$ are chosen by varying values from the set $\{10^{-5},...,10^{5}\}$. The value of μ is chosen from the set $\{2^{-5},...,2^{5}\}$ for all the algorithms. For EFSVM, the value of β for the fuzzy function [27] is chosen as 0.05, l is set as 10 and k is selected from the set $\{3, 5, 7, 9, 11\}$. In SVM-RUS and TWSVM-RUS random undersampling of the majority class is performed. In SMOTE, the value of K for K-nearest neighbours (KNN) is set as 3. In case of UTSVM, the tolerance value i.e, ε is chosen from $\{0.1, 0.3, 0.5, 0.6\}$, and 10 % of training data is chosen for the number of universum data points i.e. u. For MMTSSVM and FTSVM , $\nu_1 = \nu_2$ is selected from the set $\{0.1, 0.2, ..., 0.9\}$. For choosing the optimal parameters for training the algorithms, 5-fold cross-validation technique is applied on the training set. In case of UTSVM and RUTSVM-CIL, random averaging scheme is used for the generation of universum. In RUTSVM-CIL, the minority class data points are used multiple times for random averaging of data points with the majority class.

5.2 Dataset description

We used two types of binary class imbalanced datasets i.e., synthetic and real world datasets. Three synthetic datasets are used namely, Crescent_&_full_moon, Half_kernel [36], and Crossplane (XOR) dataset [37] as shown in Figs. 2, 3 and 4 respectively. For Crescent_&_full _moon and Half_kernel datasets, total number of samples is set as 500 with 475 data points of negative class and 25 data points of positive class.

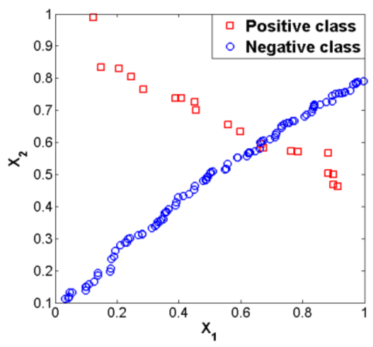


Figure 4: Crossplane dataset containing 120 samples with imbalance ratio, $\mathrm{IR} = 5$.

For Crossplane dataset, the parameters k_1 and b_1 [37] are set as 0.7 and 0.1 for majority class and, 0.6 and 1 for minority class. The imbalance ratios of Crescent_&_full_moon, Half_kernel datasets are shown in Table 1, and for Crossplane datasets in Table 3.

In order to justify the applicability of RUTSVM-CIL for real world class imbalance problems, we used 28 real world datasets from KEEL imbalanced datasets [38] and UCI repository [39]. The imbalance ratios are shown in Table 6. The imbalance ratio (IR) of 6 datasets lie in the range (2, 5], 12 datasets lie in (5, 10], 6 datasets lie in (10, 15], and 4 highly imbalanced datasets lie in (15, 33].

We also performed experiments on large scale datasets generated using the approach as mentioned in [40]. The parameters are set as $C_1 = C_2 = C_u = 1$, $\mu = 2$, and $\varepsilon = 0.5$. For FTSVM, $\nu_1 = \nu_2$ is set as 0.3. The datasets are made class imbalanced by randomly removing samples of one class.

5.3 Synthetic datasets

In order to analyse the performance of the different algorithms, we performed experiments on synthetic datasets i.e., Crescent_&_full_moon, Half_kernel, and Crossplane.

5.3.1. Effect of prior information

Figs. 2 and 3 illustrate the effect of universum in the proposed RUTSVM-CIL. The blue colour curves show the positive class hyperplane, and the curves in red are the negative class hyperplane. It is clearly visible that the classifier of the proposed RUTSVM-CIL utilizes prior information from the universum to obtain a better classifier. The classifiers in the other algorithms do not have any prior information leading to misclassification of data. Moreover, it can be seen in Figs. 2 and 3 that the classifiers of the existing algorithms are biased towards the majority class. The proposed algorithm is not showing any bias leading to better classification of imbalanced data. This is due to the universum data which creates a balance between the classes.

5.3.2 Effect of training size

In order to show the effect of training size on the performance of the various algorithms, experiments are performed on Crescent_&_full_moon and Half_kernel datasets for different sets of training data. The total number of samples in Crescent_&_full_moon and Half_kernel datasets is 500. The training data is selected as 30 %, 40 %, 50 % and 70 % of the number of samples in the datasets. The performance analysis of different algorithms is shown in Table 1 in terms of AUC values and training time. The corresponding rank table is given in Table 2 for the performance comparison of proposed RUTSVM-CIL with EFSVM, SVM-RUS, TWSVM, TWSVM-RUS, TWSVM-SMOTE, MMTSSVM, FTSVM and UTSVM. The proposed algorithm performs better than the existing algorithms in 4 out of 8 datasets with least rank for all the 8 datasets. Figs. 5 and 6 show the plot of AUC and training time of Crescent_&_full_moon and Half_kernel dataset for different training sizes.

Dataset (Train size, Test size)	IR (All samples)	IR (Training samples)	EFSVM AUC (%) (C,μ,K) Time (s)	SVM-RUS AUC (%) (C,μ) Time (s)	TWSVM AUC (%) (C,μ) Time (s)	TWSVM-RUS AUC (%) (C,μ) Time (s)	TWSVM-SMOTE AUC (%) (C,μ) Time (s)	MMTSSVM AUC (%) (ν,μ) Time (s)	FTSVM AUC (%) (ν,μ) Time (s)	UTSVM AUC (%) (C,μ,ε) Time (s)	Proposed RUTSVM-CIL AUC (%) (C,μ,ε) Time (s)
Crescent_&_full_moon (150 × 2, 350 × 2)	19	20.43	88.7383 (10^3, 2^3, 3) 0.2676	**95.8668** (10^2, 2^3) 0.0051	85.9605 (10^-3, 2^5) 0.0336	94.1432 (10^-5, 2^5) 0.0049	86.1111 (10^-2, 2^3) 0.0389	80.7564 (0.9, 2^-1) 0.0675	94.2938 (0.6, 2^5) 0.03	85.9605 (10^-3, 2^5, 0.1) 0.0356	88.2865 (10^-2, 2^2, 0.3) 0.0166
Crescent_&_full_moon (200 × 2, 300 × 2)	19	17.18	82.1429 (10^5, 2^3, 11) 0.4664	**93.2817** (10^1, 2^2) 0.0068	92.3327 (10^-5, 2^5) 0.0525	92.3327 (10^-5, 2^5) 0.0047	92.8571 (10^-3, 2^1) 0.085	70.8292 (0.5, 2^-1) 0.0505	92.3327 (0.6, 2^5) 0.0475	89.2857 (10^0, 2^5, 0.6) 0.0624	92.6823 (10^-1, 2^2, 0.5) 0.0209
Crescent_&_full_moon (250 × 2, 250 × 2)	19	19.83	80.7692 (10^5, 2^3, 7) 0.7337	90.6199 (10^-5, 2^0) 0.0077	88.0396 (10^-5, 2^5) 0.0812	91.6748 (10^-5, 2^5) 0.0052	84.6154 (10^-4, 2^3) 0.1022	43.7845 (0.8, 2^-1) 0.0757	88.0396 (0.6, 2^5) 0.0726	88.0396 (10^-5, 2^5, 0.1) 0.0903	**95.9429** (10^0, 2^3, 0.3) 0.0313
Crescent_&_full_moon (350 × 2, 150 × 2)	19	18.44	92.8571 (10^5, 2^5, 11) 1.4395	79.2208 (10^-5, 2^-3) 0.0146	92.1578 (10^-5, 2^5) 0.1585	81.8681 (10^-5, 2^4) 0.0073	92.5075 (10^1, 2^3) 0.2085	71.1788 (0.1, 2^-1) 0.1418	85.3646 (0.4, 2^5) 0.1924	92.1578 (10^-5, 2^5, 0.1) 0.1802	**98.951** (10^0, 2^5, 0.3) 0.0591
Half_kernel (150 × 2, 350 × 2)	19	20.43	**97.2222** (10^2, 2^4, 3) 0.2645	87.8012 (10^3, 2^5) 0.0065	58.0321 (10^-4, 2^5) 0.0321	84.5884 (10^-2, 2^4) 0.0053	83.3333 (10^-5, 2^2) 0.0399	96.921 (0.1, 2^1) 0.0322	69.4444 (0.6, 2^1) 0.0302	93.0054 (10^3, 2^5, 0.1) 0.0391	86.1111 (10^-1, 2^4, 0.5) 0.0121
Half_kernel (200 × 2, 300 × 2)	19	19	93.3333 (10^3, 2^5, 3) 0.4644	96.1404 (10^2, 2^5) 0.0077	93.3333 (10^-2, 2^4) 0.0552	86.6667 (10^-2, 2^5) 0.0059	**96.6667** (10^-5, 2^4) 0.0667	96.4912 (0.1, 2^1) 0.0526	90 (0.1, 2^4) 0.0627	79.8246 (10^-1, 2^1, 0.6) 0.0633	**96.6667** (10^0, 2^5, 0.1) 0.0234
Half_kernel (250 × 2, 250 × 2)	19	19.83	92.3077 (10^2, 2^4, 9) 0.7261	94.7257 (10^1, 2^4) 0.01	88.4615 (10^-2, 2^4) 0.0924	86.7089 (10^-1, 2^4) 0.0069	92.3077 (10^-5, 2^4) 0.1035	96.1538 (0.2, 2^1) 0.081	92.3077 (0.2, 2^4) 0.0983	88.0396 (10^-2, 2^0, 0.5) 0.106	**98.3122** (10^1, 2^5, 0.5) 0.0299
Half_kernel (350 × 2, 150 × 2)	19	17.42	91.6667 (10^2, 2^5, 3) 1.4462	**96.875** (10^1, 2^4) 0.0162	91.6667 (10^-3, 2^4) 0.1754	91.6667 (10^0, 2^5) 0.01	91.6667 (10^-5, 2^4) 0.2073	90.2778 (0.1, 2^2) 0.1839	91.6667 (0.1, 2^4) 0.192	91.6667 (10^-3, 2^4, 0.1) 0.1854	94.7917 (10^1, 2^5, 0.2) 0.0606

Table 1: AUC values and training time of the algorithms for different sets of training data for classification on synthetic class imbalanced datasets. IR represents the imbalance ratio of the dataset, time is in seconds.

Dataset	EFSVM	SVM-RUS	TWSVM	TWSVM-RUS	TWSVM-SMOTE	MMTSSVM	FTSVM	UTSVM	Proposed RUTSVM-CIL
Crescent_&_full_moon	4	1	7.5	3	6	9	2	7.5	5
Crescent_&_full_moon	8	1	5	5	2	9	5	7	3
Crescent_&_full_moon	8	3	5	2	7	9	5	5	1
Crescent_&_full_moon	2	8	4.5	7	3	9	6	4.5	1
Half_kernel	1	4	9	6	7	2	8	3	5
Half_kernel	5.5	4	5.5	8	1.5	3	7	9	1.5
Half_kernel	5	3	7	9	5	2	5	8	1
Half_kernel	5.5	1	5.5	5.5	5.5	9	5.5	5.5	2
Average rank	4.875	3.125	6.125	5.6875	4.625	6.5	5.4375	6.1875	**2.4375**

Table 2: Comparison of rank of RUTSVM-CIL on the basis of AUC with existing algorithms on different sets of training data for classification on synthetic class imbalanced datasets.

One can observe in Figs. 5(a) and 6(a) that the AUC of the proposed RUTSVM-CIL is increasing with increase in training size. There is a slight decrease in the AUC of RUTSVM-CIL in Fig. 6(a) with 70 % training data. This may be attributed to the selection of universum using random averaging of data points. Figs. 5(b) and 6(b) show the comparison of training time of the various algorithms except EFSVM, since it is having very high computation time in comparison to the other algorithms. It is visible that the training time of RUTSVM-CIL is lesser than most of the algorithms. Also, the rate of increase in training time of RUTSVM-CIL is lesser than other algorithms.

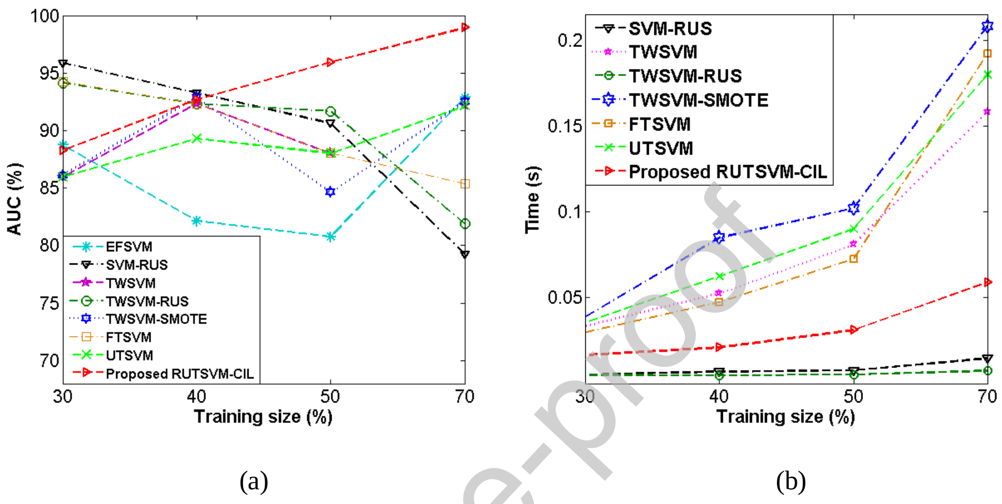


Figure 5: (a) Plot of AUC vs training size and (b) Time vs training size for Crescent_&_full_moon dataset with 500 samples.

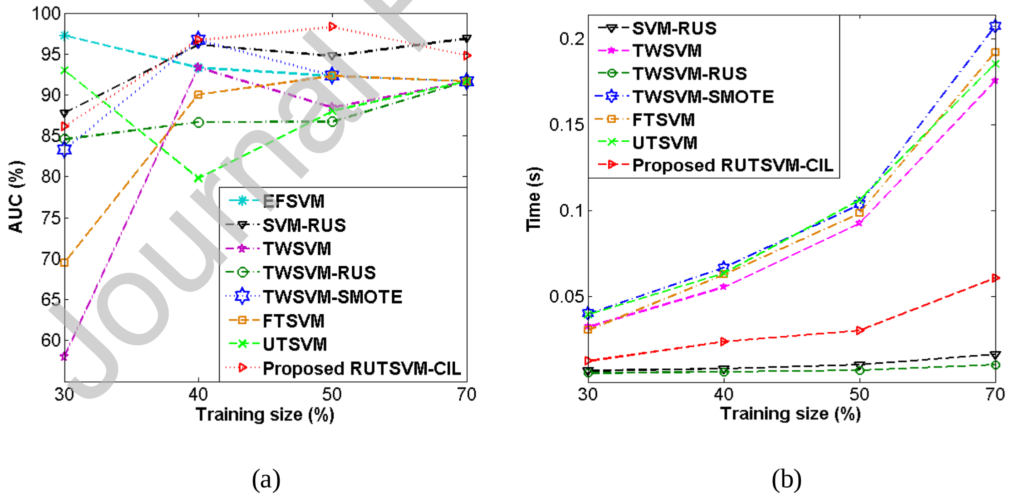


Figure 6: (a) Plot of AUC vs training size and (b) Time vs training size for Half_kernel dataset with 500 samples.

5.3.3 Crossplane dataset

Numerical experiments are performed on synthetic Crossplane dataset to verify the effectiveness of the proposed RUTSVM-CIL. Table 3 shows the AUC and training of the different algorithms on Crossplane dataset. RUTSVM-CIL performs better than the existing algorithms with lesser training time in most cases. Moreover, RUTSVM-CIL obtains least rank in Table 4.

Dataset (Train size, Test size)	IR (All samples)	EFSVM AUC (%) (C, μ, K) Time (s)	SVM-RUS AUC (%) (C, μ) Time (s)	TWSVM AUC (%) (C, μ) Time (s)	TWSVM-RUS AUC (%) (C, μ) Time (s)	TWSVM-SMOTE AUC (%) (C, μ) Time (s)	MMTSSVM AUC (%) (ν, μ) Time (s)	FTSVM AUC (%) (ν, μ) Time (s)	UTSVM AUC (%) (C, μ, ε) Time (s)	Proposed RUTSVM-CIL AUC (%) (C, μ, ε) Time (s)
Crossplane_120 (50 × 2, 70 × 2)	5	88.4615 (10^-5, 2^-4, 3) 0.0339	89.4737 (10^-5, 2^-5) 0.0042	91.4305 (10^-5, 2^1) 0.0108	**98.2456** (10^-5, 2^-1) 0.0045	97.3684 (10^-5, 2^-3) 0.0096	68.0837 (10^0.1, 2^-5) 0.0077	91.4305 (0.6, 2^1) 0.0074	91.4305 (10^-5, 2^1, 0.1) 0.0085	95.2767 (10^-5, 2^0, 0.1) 0.0076
Crossplane_250 (150 × 2, 100 × 2)	4	93.1818 (10^-5, 2^-5, 3) 0.3082	95.5128 (10^-5, 2^-5) 0.0674	94.1725 (10^-2, 2^2) 0.0791	95.1632 (10^-5, 2^1) 0.0533	98.7179 (10^-5, 2^-5) 0.1207	75.4079 (0.1, 2^-1) 0.0701	95.4545 (0.3, 2^-1) 0.0896	97.0862 (10^-2, 2^-4, 0.1) 0.0763	**99.359** (10^-4, 2^-4, 0.6) 0.0692
Crossplane_330 (110 × 2, 220 × 2)	10	95.8333 (10^-5, 2^-5, 3) 0.1432	88.2653 (10^-5, 2^-5) 0.0037	97.4065 (10^-3, 2^1) 0.0252	96.6837 (10^-5, 2^0) 0.0047	97.1514 (10^-5, 2^1) 0.0222	97.9167 (10^0.1, 2^-5) 0.0195	92.7296 (0.4, 2^0) 0.0208	97.4065 (10^-3, 2^1, 0.5) 0.0207	**99.2347** (10^-5, 2^1, 0.1) 0.0086
Crossplane_400 (133 × 2, 267 × 2)	3	95.2381 (10^0, 2^-5, 3) 0.3185	98.4711 (10^1, 2^-5) 0.091	97.3739 (10^-4, 2^-2) 0.0624	**99.5098** (10^-5, 2^-4) 0.0561	98.9613 (10^-4, 2^-2) 0.1167	51.3539 (0.1, 2^-4) 0.0633	**99.5098** (0.3, 2^-2) 0.0857	98.9613 (10^-2, 2^-2, 0.2) 0.1279	98.4127 (10^-3, 2^-2, 0.6) 0.0616

Table 3: AUC and training time of proposed RUTSVM-CIL and existing algorithms for classification of synthetic Crossplane imbalanced datasets.

Dataset	EFSVM	SVM-RUS	TWSVM	TWSVM-RUS	TWSVM-SMOTE	MMTSSVM	FTSVM	UTSVM	Proposed RUTSVM-CIL
Crossplane_120	8	7	5	1	2	9	5	5	3
Crossplane_250	8	4	7	6	2	9	5	3	1
Crossplane_330	7	9	3.5	6	5	2	8	3.5	1
Crossplane_400	8	5	7	1.5	3.5	9	1.5	3.5	6
Average rank	7.75	6.25	5.625	3.625	3.125	7.25	4.875	3.75	**2.75**

Table 4: Comparison of rank of RUTSVM-CIL on the basis of AUC with existing algorithms for classification on synthetic Crossplane class imbalanced datasets.

5.3.4 Effect of imbalance ratio (IR)

To verify the efficacy of the reduced kernel, the performance comparison is made using Crossplane dataset in Fig. 7. It is clear from Fig. 7 that the proposed RUTSVM-CIL gives better accuracy with less training time for highly imbalanced data. However, the accuracy of RUTSVM-CIL is less for $IR = 4$. This is due to the use of reduced kernel in proposed RUTSVM-CIL, resulting into removal of some informative data points. Fig. 8 justifies this fact where the accuracy of proposed RUTSVM-CIL with full kernel matrix is higher than RUTSVM-CIL with reduced kernel for $IR = 4$. For higher imbalance ratio, less number of samples is used in the construction of reduced kernel matrix. Moreover, the decline in accuracy of RUTSVM-CIL is not so rapid in comparison to other algorithms. Most of the existing algorithms are having fluctuations in their accuracies, while RUTSVM-CIL is rather stable w.r.t. its declining accuracy values for higher IRs. To check the effect of reduced kernel, performance of RUTSVM-CIL is compared with proposed scheme using full kernel matrix and UTSVM in Fig. 8.

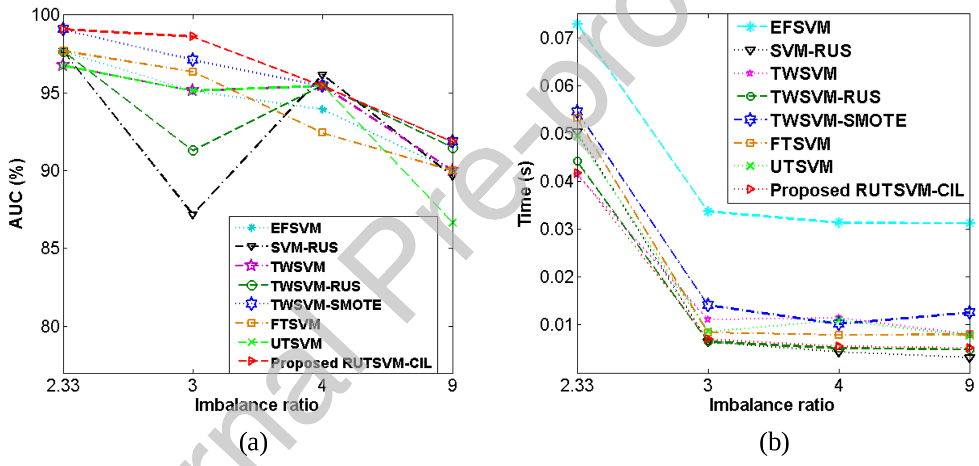


Figure 7: (a) Plot of AUC vs imbalance ratio (IR) and (b) Time vs IR for Crossplane dataset containing 200 samples with training size of 50 and testing size of 150 data points.

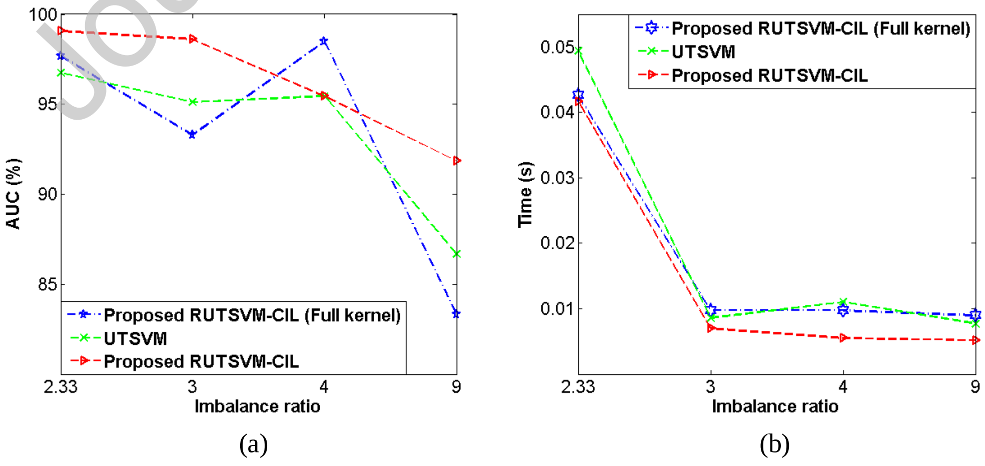


Figure 8: (a) Plot of AUC vs imbalance ratio (IR) and (b) Time vs IR for Crossplane dataset with training size of 50 and testing size of 150 data points.

It is clearly visible that the proposed RUTSVM-CIL is giving better generalization performance in most cases as compared to proposed algorithm with full kernel matrix. Moreover, the training time of the proposed algorithm is also very less.

5.3.5 Selection of universum

In the formulation of proposed RUTSVM-CIL, the matrix U^* is a subset of universum matrix U of size equal to ceiling of $\frac{r}{2}$, where r is number of data points of minority class. The data points in U^* are selected randomly from U. To analyze the effect of random selection of U^* on the generalization performance, we conducted experiments on three synthetic datasets i.e., Crescent_&_full_moon, Half_kernel, and Crossplane. The training size is set as 50 % of total samples. The performance of RUTSVM-CIL on different sizes of U^* is shown in Fig. 9, where the size of U^* is a fraction of U. One can clearly observe that on the three synthetic datasets, the performance is high when the size of U^* is 0.5 or 0.75 times the samples in minority class. This is because for higher size of U^*, less importance is given for the classification of minority class samples. So, this analysis justifies our method for proper selection of U^*.

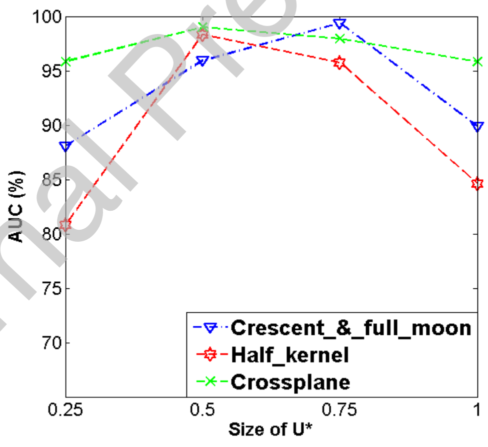


Figure 9: (a) Plot of AUC vs size of U^* on synthetic datasets. The size of U^* is represented as a fraction of the minority class samples.

5.4 Real world datasets

In this subsection, numerical experiments are performed on several real world binary class imbalanced datasets. The performance of the proposed RUTSVM-CIL is compared with EFSVM, SVM-RUS, TWSVM, TWSVM-RUS, TWSVM-SMOTE, MMTSSVM, FTSVM and UTSVM.

5.4.1 Generalization performance

The classification accuracy of the algorithms is shown in Table 5 in terms of AUC values and training time. The corresponding ranks are also shown in Table 5. It is observable that the proposed approach is showing better AUC in 8 out of 28 datasets. The proposed RUTSVM-CIL is also having the least rank on the basis of AUC i.e., 2.6607 as shown in Table 5. For 6 datasets, the rank of RUTSVM-CIL is 1 including Shuttle-c0-vs-c4, which is a large dataset. It is observable that TWSVM-RUS is having less rank than TWSVM which shows the efficacy of undersampling with TWSVM. The average rank of UTSVM is more than RUTSVM-CIL i.e., $5.375 > 2.6607$. This is due to biasing of the classifier towards majority class in UTSVM. EFSVM is also showing good generalization ability with an average rank of 4.7857 in comparison to SVM-RUS, TWSVM-SMOTE, FTSVM and UTSVM.

The average rank of the proposed RUTSVM-CIL in Table 5 is 3.1667 for datasets with IR in the range (2, 5], 2.2917 with IR in (5, 10], 2.3333 with IR in (10, 15], and 3.5 with IR in (15, 33]. Although the overall average rank of RUTSVM-CIL is better than existing algorithms, the performance is highest for datasets with IR in the range (5, 10].

For comprehensive comparison of proposed RUTSVM-CIL with existing algorithms on different sets of testing data, we calculated mean and standard deviation (SD) of AUC and g-mean [41] in Table 6. The AUC and g-mean are calculated on 5 folds of testing data. As shown in Table 6, the proposed RUTSVM-CIL is having least rank for AUC as well as g-mean. This shows the superiority of proposed RUTSVM-CIL over existing algorithms.

Fig. 9 shows the performance of proposed RUTSVM-CIL with TWSVM and UTSVM for Ecoli-0-4-6_vs_5, Shuttle-c0-vs-c4, Ecoli3 and New_thyroid2 datasets. The figure shows the distance of the data points with the two hyperplanes. Hyperplane 1 and 2 correspond to positive and negative class respectively. One can observe from Fig. 9 that the proposed RUTSVM-CIL classifier is less biased towards the negative class. The data points of the positive and negative class are relatively closer to their own class' hyperplane as compared to the other class. Moreover, it is visible from Fig. 9(a) that the distance of negative class from its hyperplane is more in case of RUTSVM-CIL in comparison to TWSVM and UTSVM. This shows the effect of universum in reducing the bias to the negative or majority class. The majority class data points which are near to the minority class are given less importance.

5.4.2 Computation time

One can observe the computation time of the different algorithms in Table 5. The training time of RUTSVM-CIL is less than all the existing algorithms except SVM-RUS and TWSVM-RUS. This is due to undersampling of the majority class in one QPP, as well as oversampling of the universum in the other QPP of RUTSVM-CIL. Also, RUTSVM-CIL finds the solution of two smaller sized QPPs with reduced kernel matrix. It can be seen in Table 5 that the training time of UTSVM is more than TWSVM due to the universum. However, in comparison to TWSVM, our RUTSVM-CIL

Dataset (Train size, Test size)	IR (All samples)	IR (Training samples)	EFSVM AUC (%) (C, μ, K) Time (s)	SVM-RUS AUC (%) (C, μ) Time (s)	TWSVM AUC (%) (C, μ) Time (s)	TWSVM-RUS AUC (%) (C, μ) Time (s)	TWSVM-SMOTE AUC (%) (C, μ) Time (s)	MMTSSVM AUC (%) (ν, μ) Time (s)	FTSVM AUC (%) (ν, μ) Time (s)	UTSVM AUC (%) (C, μ, ε) Time (s)	Proposed RUTSVM-CIL AUC (%) (C, μ, ε) Time (s)
Ecoli-0-1_vs_2-3-5 (120 × 7, 124 × 7)	9.17	9	78.7202 (10^0, 2^5, 7) 0.2099	73.8095 (10^-5, 2^1) 0.0431	75 (10^-5, 2^5) 0.0758	75.1488 (10^-5, 2^4) 0.0385	77.8274 (10^-4, 2^5) 0.063	**81.25** (0.5, 2^3) 0.0603	75 (0.2, 2^5) 0.0861	75 (10^-5, 2^5, 0.1) 0.0537	78.7202 (10^-5, 2^5, 0.1) 0.0403
Ecoli-0-1_vs_5 (120 × 6, 120 × 6)	11	16.14	84.6154 (10^-5, 2^1, 3) 0.1706	86.7002 (10^1, 2^5) 0.0052	76.9231 (10^-5, 2^5) 0.024	87.527 (10^-5, 2^5) 0.0051	88.4615 (10^-4, 2^5) 0.0476	**92.4155** (0.1, 2^4) 0.0234	80.7692 (0.3, 2^4) 0.0238	80.7692 (10^-2, 2^4, 0.4) 0.032	86.7002 (10^-1, 2^5, 0.5) 0.0112
Ecoli-0-1-4-7_vs_5-6 (150 × 6, 182 × 6)	12.28	10.54	**91.3725** (10^0, 2^5, 5) 0.2693	85.2451 (10^-5, 2^1) 0.0091	83.3333 (10^-4, 2^4) 0.0532	85.4902 (10^-5, 2^4) 0.0051	90.4902 (10^-4, 2^4) 0.0457	68.4314 (0.1, 2^3) 0.0323	87.2059 (0.4, 2^4) 0.0375	84.0196 (10^-2, 2^3, 0.5) 0.036	88.7255 (10^0, 2^5, 0.1) 0.0162
Ecoli-0-3-4-6_vs_5 (100 × 7, 105 × 7)	9.25	8.09	88.8889 (10^0, 2^5, 3) 0.121	88.3681 (10^1, 2^5) 0.007	83.3333 (10^-5, 2^5) 0.0168	**94.4444** (10^-5, 2^4) 0.0048	**94.4444** (10^-3, 2^4) 0.0245	77.7778 (0.8, 2^3) 0.0169	77.7778 (0.2, 2^5) 0.0178	88.8889 (10^-4, 2^5, 0.4) 0.0186	**94.4444** (10^-5, 2^5, 0.1) 0.0093
Ecoli-0-4-6_vs_5 (100 × 6, 103 × 6)	9.15	11.5	83.3333 (10^-1, 2^4, 3) 0.1198	87.5916 (10^-5, 2^1) 0.0049	75 (10^-4, 2^5) 0.0165	78.6172 (10^-5, 2^4) 0.0043	82.7839 (10^-4, 2^4) 0.0225	90.5678 (0.4, 2^4) 0.0176	79.1667 (0.4, 2^4) 0.0181	82.7839 (10^-2, 2^4, 0.6) 0.0294	**91.1172** (10^0, 2^5, 0.1) 0.0092
Ecoli-0-6-7_vs_5 (110 × 6, 110 × 6)	10	9	82.8383 (10^-5, 2^3, 5) 0.1466	77.5028 (10^0, 2^4) 0.0071	**87.8988** (10^-1, 2^5) 0.0194	80.088 (10^-5, 2^5) 0.0049	79.868 (10^-5, 2^5) 0.0273	79.978 (0.2, 2^3) 0.0202	82.8383 (0.1, 2^5) 0.0209	77.2827 (10^-2, 2^4, 0.6) 0.0213	87.4037 (10^0, 2^5, 0.6) 0.0107
Glass-0-1-4-6_vs_2 (100 × 9, 105 × 9)	11.06	11.5	60.7639 (10^3, 2^1, 3) 0.1223	55.2083 (10^1, 2^2) 0.0048	**73.9583** (10^-2, 2^5) 0.0208	67.3611 (10^0, 2^3) 0.0049	66.4931 (10^-3, 2^-1) 0.0189	49.6528 (0.3, 2^-3) 0.0164	68.2292 (0.8, 2^5) 0.0157	61.8056 (10^2, 2^2, 0.3) 0.0189	68.9236 (10^-2, 2^0, 0.4) 0.0085
Glass-0-1-5_vs_2 (80 × 9, 92 × 9)	9.12	7	54.2017 (10^0, 2^-1, 9) 0.0785	72.2689 (10^-5, 2^-2) 0.0061	50.1681 (10^-2, 2^4) 0.0122	66.9748 (10^-5, 2^4) 0.0053	**78.0672** (10^-2, 2^-2) 0.016	50 (0.3, 2^-5) 0.0122	48.9076 (0.7, 2^4) 0.013	69.916 (10^-5, 2^5, 0.1) 0.0126	54.5378 (10^0, 2^4, 0.6) 0.0079
Shuttle-c0-vs-c4 (900 × 9, 929 × 9)	13.87	13.52	98.3607 (10^-5, 2^5, 3) 9.5672	97.4258 (10^-5, 2^1) 0.1438	97.541 (10^-5, 2^4) 1.5047	97.541 (10^-5, 2^5) 0.0228	96.7213 (10^-5, 2^3) 1.956	99.6544 (0.7, 2^3) 1.3888	99.8272 (0.6, 2^4) 1.4231	97.541 (10^-5, 2^4, 0.1) 1.7636	**99.9424** (10^0, 2^5, 0.1) 0.6726
Yeast-0-2-5-6_vs_3-7-8-9 (500 × 8, 504 × 8)	9.14	11.5	72.3196 (10^2, 2^0, 7) 2.9608	**79.5677** (10^0, 2^-2) 0.0622	77.3624 (10^-5, 2^3) 0.3395	78.8421 (10^-2, 2^2) 0.0114	73.4622 (10^-1, 2^-2) 0.4512	64.0773 (0.1, 2^3) 0.3246	75.7799 (0.2, 2^3) 0.3966	77.25 (10^1, 2^1, 0.6) 0.4369	77.1377 (10^-1, 2^3, 0.1) 0.1405
Yeast-0-2-5-7-9_vs_3-6-8 (500 × 8, 504 × 8)	9.14	8.62	89.4408 (10^-1, 2^-3, 5) 3.0124	**91.4777** (10^-5, 2^-3) 0.1033	87.83 (10^-1, 2^3) 0.3537	85.8001 (10^-5, 2^3) 0.0165	82.1826 (10^1, 2^-3) 0.5275	64.7819 (0.9, 2^1) 0.3204	83.9518 (0.6, 2^4) 0.3872	84.2311 (10^0, 2^0, 0.2) 0.4124	86.8453 (10^-2, 2^-2, 0.5) 0.1605
Yeast-0-3-5-9_vs_7-8 (250 × 8, 256 × 8)	9.12	11.5	68.9233 (10^3, 2^-2, 7) 0.7508	69.9853 (10^4, 2^4) 0.0185	62.5664 (10^-2, 2^0) 0.0821	71.0914 (10^0, 2^1) 0.0065	55.354 (10^-4, 2^-4) 0.1054	56.2832 (0.1, 2^-4) 0.0865	61.0029 (0.7, 2^-2) 0.0747	68.2448 (10^2, 2^-2, 0.6) 0.1032	**71.3127** (10^0, 2^2, 0.6) 0.035
Ecoli-0-1-4-6_vs_5 (150 × 6, 130 × 6)	13	9.71	91.6667 (10^-5, 2^3, 3) 0.2689	97.9839 (10^-5, 2^1) 0.0099	91.6667 (10^-3, 2^5) 0.0329	**100** (10^-5, 2^4) 0.0055	99.5968 (10^-3, 2^4) 0.0459	89.2473 (0.3, 2^4) 0.0313	91.6667 (0.4, 2^4) 0.0342	91.6667 (10^-3, 2^5, 0.5) 0.0363	**100** (10^-1, 2^5, 0.3) 0.0157
Haber (200 × 3, 106 × 3)	2.78	2.57	**65.8272** (10^2, 2^5, 9) 0.4776	50 (10^-5, 2^5) 0.12	50.5926 (10^0, 2^2) 0.0508	50 (10^1, 2^5) 0.0189	45.5062 (10^-1, 2^2) 0.1203	50 (0.8, 2^3) 0.0457	60.8642 (0.6, 2^4) 0.0533	47.0617 (10^-2, 2^2, 0.3) 0.056	52.2963 (10^-3, 2^2, 0.5) 0.0414

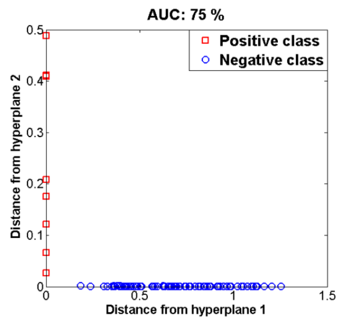
Dataset (Train size, Test size)	IR (All samples)	IR (Training samples)	EFSVM AUC (%) (C, μ, K) Time (s)	SVM-RUS AUC (%) (C, μ) Time (s)	TWSVM AUC (%) (C, μ) Time (s)	TWSVM-RUS AUC (%) (C, μ) Time (s)	TWSVM-SMOTE AUC (%) (C, μ) Time (s)	MMTSSVM AUC (%) (ν, μ) Time (s)	FTSVM AUC (%) (ν, μ) Time (s)	UTSVM AUC (%) (C, μ, ε) Time (s)	Proposed RUTSVM-CIL AUC (%) (C, μ, ε) Time (s)
Glass4 (150 × 9, 64 × 9)	15.46	17.75	**98.3051** (10^2, 2^2, 3) 0.2737	82.3729 (10^1, 2^1) 0.0049	90 (10^1, 2^4) 0.0319	64.9153 (10^-1, 2^3) 0.0045	89.1525 (10^1, 2^5) 0.0382	94.0678 (0.9, 2^0) 0.0338	89.1525 (0.1, 2^5) 0.0355	69.1525 (10^-2, 2^0, 0.5) 0.0376	89.1525 (10^-1, 2^4, 0.6) 0.0128
Ecoli3 (150 × 7, 186 × 7)	8.6	6.89	89.3382 (10^0, 2^-2, 11) 0.2708	81.9853 (10^-5, 2^-5) 0.0163	85.4044 (10^-5, 2^4) 0.0303	81.1029 (10^-3, 2^-2) 0.0061	87.5735 (10^-1, 2^-1) 0.0497	61.5441 (0.1, 2^2) 0.0298	85.9926 (0.6, 2^4) 0.0291	85.4044 (10^-5, 2^4, 0.1) 0.0357	**93.6397** (10^0, 2^1, 0.6) 0.0176
Abalone9-18 (350 × 7, 381 × 7)	16.4	18.44	76.3831 (10^5, 2^2, 7) 1.4868	85.2241 (10^3, 2^2) 0.0178	70.5532 (10^-2, 2^0) 0.1733	**85.6268** (10^-2, 2^1) 0.0061	68.0497 (10^0, 2^-2) 0.1957	50.8578 (0.9, 2^4) 0.1638	76.6632 (0.2, 2^0) 0.1929	80.8298 (10^-1, 2^0, 0.5) 0.1956	82.7556 (10^1, 2^0, 0.1) 0.063
Vehicle 1 (400 × 18, 446 × 18)	2.9	3.3	67.1409 (10^1, 2^5, 7) 1.915	70.9126 (10^2, 2^5) 0.3303	63.8249 (10^-5, 2^5) 0.2115	**71.2558** (10^-3, 2^5) 0.0462	54.711 (10^-4, 2^3) 0.5114	53.7267 (0.7, 2^3) 0.1584	65.0646 (0.8, 2^5) 0.1933	66.9881 (10^-2, 2^5, 0.3) 0.2445	69.3974 (10^1, 2^5, 0.5) 0.1581
Vehicle2 (400 × 18, 446 × 18)	2.88	2.48	**97.8148** (10^1, 2^5, 11) 1.9379	89.7492 (10^2, 2^5) 0.51	92.7666 (10^-2, 2^5) 0.2095	89.9941 (10^-2, 2^5) 0.0693	95.5306 (10^-2, 2^5) 0.682	78.2997 (0.7, 2^3) 0.1516	91.6499 (0.2, 2^5) 0.2198	97.1355 (10^-1, 2^5, 0.5) 0.2454	92.0858 (10^1, 2^5, 0.4) 0.1843
Yeast3 (500 × 8, 984 × 8)	8.1	7.2	79.6519 (10^1, 2^-3, 5) 3.0297	86.618 (10^0, 2^-3) 0.1444	**91.5433** (10^-2, 2^2) 0.3412	87.7951 (10^0, 2^1) 0.0209	80.2354 (10^-2, 2^-4) 0.578	72.4523 (0.3, 2^-5) 0.2845	89.4324 (0.3, 2^2) 0.3592	84.4204 (10^1, 2^-1, 0.5) 0.4155	91.5033 (10^-1, 2^2, 0.1) 0.1511
Yeast1 (500 × 8, 2468 × 8)	2.46	2.97	70.4751 (10^0, 2^-3, 11) 3.0056	69.6676 (10^1, 2^-3) 0.6152	71.0055 (10^-5, 2^3) 0.3103	69.5311 (10^-1, 2^-1) 0.0773	**72.7832** (10^-2, 2^-3) 0.8385	48.2912 (0.1, 2^5) 0.27	71.0055 (0.7, 2^3) 0.2903	71.0055 (10^-5, 2^3, 0.1) 0.3539	71.8316 (10^-5, 2^3, 0.1) 0.2293
Yeast1vs7 (200 × 8, 259 × 8)	14.3	15.67	58.829 (10^2, 2^-2, 3) 0.4886	**74.9424** (10^1, 2^-1) 0.0139	71.012 (10^-3, 2^2) 0.059	66.3555 (10^0, 2^2) 0.0052	57.7916 (10^1, 2^-1) 0.07	65.5717 (0.1, 2^-1) 0.0539	62.3098 (0.4, 2^1) 0.0631	63.4278 (10^1, 2^-1, 0.5) 0.0642	72.5911 (10^0, 2^2, 0.5) 0.0211
Yeast-2_vs_4 (250 × 8, 264 × 8)	9.08	8.62	82.1172 (10^5, 2^1, 3) 0.7617	87.5146 (10^2, 2^0) 0.0284	90.6527 (10^-2, 2^2) 0.0818	88.4937 (10^-1, 2^0) 0.0074	84.3264 (10^1, 2^-2) 0.1281	62.2594 (0.9, 2^2) 0.0763	91.2803 (0.7, 2^2) 0.0891	90.3515 (10^1, 2^0, 0.4) 0.0956	**92.0251** (10^0, 2^2, 0.3) 0.0362
Yeast2vs8 (250 × 8, 233 × 8)	23.15	19.83	68.75 (10^2, 2^-2, 7) 0.7468	69.8889 (10^2, 2^1) 0.0081	74.3333 (10^-5, 2^2) 0.0805	68.5556 (10^0, 2^1) 0.0051	73 (10^1, 2^-2) 0.1032	54.1667 (0.3, 2^-4) 0.0843	**74.7778** (0.7, 2^2) 0.0975	**74.7778** (10^0, 2^-1, 0.3) 0.112	74.5556 (10^-2, 2^1, 0.5) 0.0282
Ecoli0137vs26 (180 × 7, 131 × 7)	4.76	4.63	**97.7273** (10^-5, 2^-3, 3) 0.3855	89.9291 (10^-5, 2^-4) 0.0421	94.5371 (10^-5, 2^4) 0.0416	93.5988 (10^-2, 2^-2) 0.0104	92.6814 (10^-3, 2^-3) 0.087	59.2994 (0.1, 2^-1) 0.0389	93.1818 (0.1, 2^0) 0.05	93.1818 (10^2, 2^4, 0.5) 0.0491	94.0784 (10^0, 2^4, 0.1) 0.0266
Yeast5 (500 × 8, 984 × 8)	32.73	34.71	72.8616 (10^1, 2^-3, 7) 3.0079	91.8449 (10^1, 2^-3) 0.01	95.87 (10^-5, 2^3) 0.3425	93.1656 (10^-5, 2^-2) 0.0072	94.0042 (10^0, 2^0) 0.4159	64.9371 (0.1, 2^1) 0.3559	**96.5514** (0.1, 2^3) 0.3915	83.7421 (10^2, 2^0, 0.2) 0.4873	95.4507 (10^0, 2^3, 0.1) 0.1081
Wpbc (100 × 33, 94 × 33)	3.22	2.85	57.8378 (10^-5, 2^0, 9) 0.1259	**69.1216** (10^1, 2^2) 0.0289	61.7568 (10^-2, 2^5) 0.0163	63.2432 (10^-2, 2^2) 0.0078	50.2703 (10^-2, 2^4) 0.0263	55.8108 (0.9, 2^2) 0.0153	58.5811 (0.7, 2^5) 0.0166	61.7568 (10^-2, 2^5, 0.1) 0.0179	61.8919 (10^0, 2^4, 0.4) 0.0136
New_thyroid2 (90 × 5, 125 × 5)	5.14	4	91.1765 (10^0, 2^2, 3) 0.1007	96.1329 (10^2, 2^4) 0.0161	96.5959 (10^-5, 2^5) 0.0142	95.3704 (10^-2, 2^4) 0.0079	93.6547 (10^-3, 2^5) 0.0273	84.2857 (0.4, 2^1) 0.0242	90.7135 (0.8, 2^5) 0.0154	96.5959 (10^-5, 2^5, 0.1) 0.0238	**98.6111** (10^0, 2^5, 0.3) 0.0119
Average rank			4.7857	4.9821	4.6429	4.625	5.5536	7.2321	5.1429	5.375	**2.6607**

Table 5: Comparison of proposed RUTSVM-CIL on AUC and training time with existing algorithms for classification on real world imbalanced datasets.

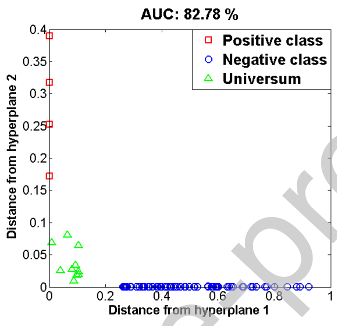
Dataset	IR	EFSVM AUC ± SD (%) G-mean ± SD (%)	SVM-RUS AUC ± SD (%) G-mean ± SD (%)	TWSVM AUC ± SD (%) G-mean ± SD (%)	TWSVM-RUS AUC ± SD (%) G-mean ± SD (%)	TWSVM-SMOTE AUC ± SD (%) G-mean ± SD (%)	MMTSSVM AUC ± SD (%) G-mean ± SD (%)	FTSVM AUC ± SD (%) G-mean ± SD (%)	UTSVM AUC ± SD (%) G-mean ± SD (%)	Proposed RUTSVM-CIL AUC ± SD (%) G-mean ± SD (%)
Ecoli-0-1_vs_2-3-5	9.17	77.42 ± 21.94 65.23 ± 40.63	**85.96 ± 14.23** **84.11 ± 17.39**	67.42 ± 21.77 45.69 ± 44.43	81 ± 7.87 78.21 ± 10.14	83.22 ± 22.22 72.79 ± 42.37	68.74 ± 18.41 47.92 ± 43.91	65.76 ± 24.39 36.33 ± 50.17	72.42 ± 26.33 51.55 ± 50.12	70.28 ± 22.56 50.08 ± 46.93
Ecoli-0-1_vs_5	11	78.71 ± 18.02 87.33 ± 12.01	81.31 ± 19.06 90.9 ± 5.65	76.25 ± 18.84 84.19 ± 14.21	81.42 ± 23.17 **91.09 ± 6.83**	**82.08 ± 21.73** 91.04 ± 12.65	76.13 ± 24.87 84.68 ± 8.34	75.83 ± 19.18 83.01 ± 18.58	77.38 ± 18.97 85.29 ± 16.82	79.78 ± 17.47 89.21 ± 8.29
Ecoli-0-1-4-7_vs_5-6	12.28	83.33 ± 20.41 92.66 ± 10.05	78.2 ± 20.3 87.96 ± 8.38	76.67 ± 18.07 84.61 ± 14.74	77.93 ± 18.21 86.48 ± 11.6	76.34 ± 18.12 84.38 ± 15.02	69.02 ± 25.05 68.1 ± 38.19	76.34 ± 18.12 84.38 ± 15.02	60 ± 22.36 40 ± 54.77	**83.79 ± 21.3** **93.51 ± 9.09**
Ecoli-0-3-4-6_vs_5	9.25	81.47 ± 20.68 90.58 ± 13.36	79.41 ± 27.25 88.56 ± 13.47	76.47 ± 17.91 84.72 ± 14.56	76.82 ± 17.31 84.96 ± 13.54	79.52 ± 21.78 87.8 ± 15.63	50 ± 0 20 ± 44.72	82.5 ± 20.92 91.46 ± 12.97	81.97 ± 21.18 91.09 ± 13.73	**83.95 ± 22.26** **93.23 ± 13.69**
Ecoli-0-4-6_vs_5	9.15	84.44 ± 14.25 81.46 ± 17.81	87.84 ± 10.91 87.48 ± 11.14	84.44 ± 15.04 81.56 ± 18.51	**88.51 ± 8.3** **87.6 ± 9.02**	88.36 ± 14.38 86.32 ± 17.97	63.77 ± 14.99 44.7 ± 32.21	86.11 ± 19.04 82.77 ± 23.6	73.45 ± 19.73 63.01 ± 37.95	85.03 ± 18.09 81.67 ± 22.63
Ecoli-0-6-7_vs_5	10	73 ± 16.9 60.84 ± 36.21	72.14 ± 14.02 70.87 ± 13.94	59 ± 22.95 20 ± 44.72	83.12 ± 9.5 81.79 ± 10.23	**87.52 ± 13.83** **86.06 ± 15.72**	63.07 ± 20.09 32.66 ± 45.38	64 ± 21.91 33.42 ± 47.21	73.5 ± 24.47 64.66 ± 39.83	78.1 ± 18.23 77.46 ± 18.34
Glass-0-1-4-6_vs_2	11.06	55.42 ± 12.07 45.38 ± 43.12	40.02 ± 17.63 33.18 ± 21.15	49.98 ± 18.43 55.42 ± 14.01	58.79 ± 17.96 59.74 ± 15.32	62.43 ± 20.48 61.44 ± 36.75	54.52 ± 11.49 33.66 ± 47.06	56.05 ± 17.98 59.1 ± 9.32	**63.91 ± 24.01** **72.05 ± 15**	55.87 ± 23.44 58.78 ± 38.81
Glass-0-1-5_vs_2	9.12	49.47 ± 1.18 19.47 ± 43.53	57.16 ± 23.72 63.01 ± 13.6	60.19 ± 32.19 59.91 ± 38.41	51.62 ± 25.4 53.78 ± 13.38	**66.43 ± 23.17** **66.65 ± 39.01**	50 ± 0 20 ± 44.72	52.71 ± 7.92 31.55 ± 45.71	44.53 ± 21.7 35.25 ± 32.35	46.33 ± 25.52 29.3 ± 22.59
Shuttle-c0-vs-c4	13.87	90.9 ± 3.95 90.36 ± 4.37	97.69 ± 3.37 97.61 ± 3.5	97.75 ± 3.41 97.67 ± 3.55	**99.94 ± 0.13** **99.94 ± 0.13**	95.73 ± 2.84 95.6 ± 2.95	99.48 ± 0.43 99.48 ± 0.44	99.88 ± 0.16 99.88 ± 0.16	97.75 ± 3.41 97.67 ± 3.55	**99.94 ± 0.13** **99.94 ± 0.13**
Yeast-0-2-5-6_vs_3-7-8-9	9.14	64.99 ± 5.44 55.83 ± 9.39	**77.83 ± 8.4** **76.75 ± 10**	73.23 ± 8.09 69.35 ± 12.34	72.21 ± 9.49 69.02 ± 13.47	75.09 ± 6.01 73.04 ± 7.97	63.21 ± 3.45 57.98 ± 6.81	66.65 ± 6.8 57.62 ± 13.2	74.42 ± 5.92 71.73 ± 8.98	77.11 ± 8.68 75.05 ± 11.52
Yeast-0-2-5-7-9_vs_3-6-8	9.14	91.13 ± 3.3 90.73 ± 3.58	93.09 ± 2.35 93.04 ± 2.3	86.3 ± 5.44 85.14 ± 6.62	**94.29 ± 1.79** **94.24 ± 1.79**	88.48 ± 1.67 87.95 ± 1.77	54.87 ± 4.98 40.31 ± 8.49	88.24 ± 7.97 87.47 ± 8.77	93.21 ± 3.31 93.04 ± 3.44	90.27 ± 3.18 89.93 ± 3.44
Yeast-0-3-5-9_vs_7-8	9.12	55.81 ± 5.95 47.36 ± 7.72	**67.44 ± 2.5** **66.92 ± 2.83**	58.45 ± 6.08 38.97 ± 22.68	67.13 ± 8.48 64.67 ± 9.57	53.41 ± 9.7 30.18 ± 29.32	52.92 ± 7.58 37.09 ± 7.2	59.33 ± 6.2 39.4 ± 22.86	56.19 ± 10.96 43.87 ± 26.42	64.19 ± 10.28 61.37 ± 9.17
Ecoli-0-1-4-6_vs_5	13	**100 ± 0** **100 ± 0**	93.88 ± 5.33 93.54 ± 5.82	95 ± 11.18 94.14 ± 13.1	96.4 ± 8.05 96 ± 8.94	**100 ± 0** **100 ± 0**	**100 ± 0** **100 ± 0**	95 ± 11.18 94.14 ± 13.1	**100 ± 0** **100 ± 0**	99.6 ± 0.89 99.6 ± 0.9
Haber	2.78	56.49 ± 7.91 39.71 ± 23.82	50 ± 0 0 ± 0	56.21 ± 10.79 30.27 ± 31.3	44.18 ± 21.95 13.87 ± 31.01	**67.26 ± 12.4** **63.96 ± 13.32**	51.39 ± 5.62 26.92 ± 27.61	66.83 ± 14.22 59.51 ± 22.25	67.07 ± 19.53 55.78 ± 35.68	62.74 ± 17.27 48.74 ± 33.01
Glass4	15.46	66.52 ± 22.62 56.35 ± 51.57	76.34 ± 22.87 85.4 ± 13.99	56.67 ± 19.9 38.26 ± 52.48	80.13 ± 23.01 89.05 ± 10.9	81.67 ± 19.9 90.66 ± 11.97	70 ± 27.39 60 ± 54.77	81.59 ± 19.61 90.62 ± 11.55	**87.49 ± 23.21** **97.43 ± 2.36**	79.17 ± 26.68 79.15 ± 44.28
Ecoli3	8.6	84.61 ± 22.4 74.96 ± 42.72	80.92 ± 10.33 80.55 ± 10.3	82.37 ± 9.02 81.84 ± 9.07	82.38 ± 9.43 81.94 ± 9.54	70.88 ± 22.86 51.98 ± 47.69	50.8 ± 8.23 23.71 ± 16.07	**85.41 ± 8.24** **84.85 ± 8.32**	82.37 ± 9.02 81.84 ± 9.07	82.61 ± 9.82 82.18 ± 9.97
Abalone9-18	16.4	67.62 ± 10.73 73.95 ± 15.92	**79 ± 19.56** **88.36 ± 9.37**	71.19 ± 18.71 77.26 ± 22.05	70.96 ± 18.05 78.96 ± 16.08	67.53 ± 13.09 73.76 ± 17.65	45.76 ± 18.15 39.1 ± 32.45	76.07 ± 19.54 83.3 ± 20.25	66.41 ± 10.91 71.67 ± 17.35	69.5 ± 20.24 68.29 ± 40.54

Dataset	IR	EFSVM AUC ± SD (%) G-mean ± SD (%)	SVM-RUS AUC ± SD (%) G-mean ± SD (%)	TWSVM AUC ± SD (%) G-mean ± SD (%)	TWSVM-RUS AUC ± SD (%) G-mean ± SD (%)	TWSVM-SMOTE AUC ± SD (%) G-mean ± SD (%)	MMTSSVM AUC ± SD (%) G-mean ± SD (%)	FTSVM AUC ± SD (%) G-mean ± SD (%)	UTSVM AUC ± SD (%) G-mean ± SD (%)	Proposed RUTSVM-CIL AUC ± SD (%) G-mean ± SD (%)
Vehicle 1	2.9	**69.96 ± 3.23** 68.64 ± 4.26	69.64 ± 5.59 **69.05 ± 4.95**	66.6 ± 5.59 64.24 ± 6.83	65.96 ± 4.93 65.56 ± 5.02	63.57 ± 5.52 59.44 ± 9.2	57.99 ± 1.48 46.87 ± 7.44	66.96 ± 6.14 64.92 ± 7.37	67.87 ± 4.59 65.73 ± 5.99	67.74 ± 4.22 67.55 ± 4.27
Vehicle2	2.88	**92.96 ± 3.46** **92.86 ± 3.52**	92.77 ± 3.13 92.64 ± 3.15	85.38 ± 6.69 84.04 ± 8.23	84.37 ± 5.78 84.01 ± 6.19	90.43 ± 2.7 90.28 ± 2.72	73.18 ± 7.59 67.89 ± 11.53	84.83 ± 5.25 83.65 ± 6.47	91.2 ± 3.27 90.94 ± 3.48	84.89 ± 6.22 83.85 ± 7.94
Yeast3	8.1	79.94 ± 4.81 77.94 ± 6.27	**90.53 ± 4.53** **90.48 ± 4.52**	86.51 ± 4.98 85.92 ± 5.42	89.31 ± 4.92 89.07 ± 4.99	80.85 ± 5.56 79.45 ± 6.29	66.36 ± 6.04 59.32 ± 11.45	85.27 ± 4.82 84.39 ± 5.41	88.25 ± 2.8 87.98 ± 2.95	89.84 ± 4.5 89.71 ± 4.52
Yeast1	2.46	75.24 ± 2.9 74.04 ± 3.29	75.33 ± 6.67 74 ± 6.34	58.44 ± 0.76 42.55 ± 2.09	72 ± 0.85 70.32 ± 0.94	**81.3 ± 7.73** **80.83 ± 7.49**	53.9 ± 2.76 46.4 ± 5.42	71.61 ± 1.23 71.56 ± 1.24	58.44 ± 0.76 42.55 ± 2.09	58.77 ± 3.19 42.58 ± 8.05
Yeast1vs7	14.3	**70.21 ± 22.98** 50.55 ± 47.19	65.24 ± 20.77 63.69 ± 21.36	68.92 ± 18.56 **65.48 ± 21.77**	69.41 ± 17.83 64.14 ± 21.26	66.68 ± 12.84 61.12 ± 17.26	67.6 ± 8.96 64.22 ± 7.22	53.97 ± 6.88 19.79 ± 27.1	67.46 ± 17.94 54.32 ± 35.05	65.72 ± 19.64 61.38 ± 22.13
Yeast-2_vs_4	9.08	85.98 ± 8.98 84.69 ± 10.46	85.4 ± 9.33 84.33 ± 10.74	87.75 ± 15.19 85.18 ± 20.46	87.94 ± 5.9 87.22 ± 6.34	83.9 ± 13.92 80.99 ± 18.79	58.13 ± 10.25 46.35 ± 16.91	87.15 ± 15.03 84.57 ± 20.26	88.6 ± 9.45 87.67 ± 10.98	**92.96 ± 5.03** **92.68 ± 5.34**
Yeast2vs8	23.15	58.12 ± 11.98 65.48 ± 40.88	60.08 ± 13.8 69.12 ± 40.99	61.67 ± 16.24 70.47 ± 41.34	54.56 ± 15.69 66.74 ± 40.45	59.34 ± 22.13 59.57 ± 54.38	33.05 ± 26.26 33.07 ± 19.31	61.67 ± 16.24 70.47 ± 41.34	61.67 ± 16.24 70.47 ± 41.34	**63.95 ± 19.65** **73.44 ± 41.8**
Ecoli0137vs26	4.76	**98.35 ± 2.71** **98.3 ± 2.8**	95.27 ± 7.26 95.22 ± 7.32	92.09 ± 10.82 91.16 ± 12.56	95.12 ± 2.49 94.96 ± 2.63	96.09 ± 3.58 96.02 ± 3.62	56.48 ± 4.03 36.82 ± 14.83	96.25 ± 8.39 95.81 ± 9.37	96.33 ± 4.65 96.31 ± 4.64	83.8 ± 17.47 79.72 ± 22.95
Yeast5	32.73	87.94 ± 7.32 86.93 ± 8.11	96.54 ± 1.41 96.47 ± 1.46	**98 ± 1.05** **97.98 ± 1.07**	95.99 ± 1.66 95.92 ± 1.69	94.05 ± 7.46 93.65 ± 8.12	88.23 ± 2.69 87.6 ± 2.72	97.8 ± 1.19 97.77 ± 1.21	89.55 ± 5.64 88.87 ± 6.21	96.7 ± 1.9 96.63 ± 1.97
Wpbc	3.22	37.8 ± 4.73 18.35 ± 41.04	39.6 ± 18.06 47.16 ± 15.6	52.68 ± 23.46 58.15 ± 10.07	41.37 ± 20.37 43.63 ± 7.4	44.94 ± 18.67 40.58 ± 23.77	38.3 ± 21.72 8.26 ± 18.46	52.68 ± 23.46 58.15 ± 10.07	52.68 ± 23.46 58.15 ± 10.07	**56.12 ± 18.74** **64.58 ± 18.05**
New_thyroid2	5.14	81.21 ± 20.73 70.1 ± 41.22	94.19 ± 8.26 94.09 ± 8.35	85 ± 14.91 82.02 ± 18.47	94.42 ± 4.59 94.15 ± 4.9	90.36 ± 7.95 89.5 ± 8.77	82.92 ± 4.88 80.97 ± 5.94	84.52 ± 14.34 81.54 ± 17.9	85 ± 14.91 82.02 ± 18.47	**95.74 ± 8.27** **95.47 ± 8.84**
Average rank (AUC)		5.1964	4.3929	5.6964	4.1964	4.4643	7.8036	4.9286	4.5179	**3.8036**
Average rank (G-mean)		5.6607	3.9286	5.7321	3.875	4.7143	7.5536	5.1786	4.5179	**3.8393**

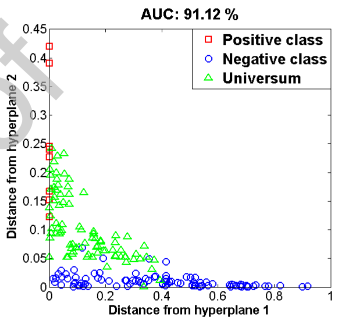
Table 6: Comparison of RUTSVM-CIL with existing algorithms on mean and standard deviation (SD) of AUC and g-mean for classification of imbalanced datasets.



TWSVM

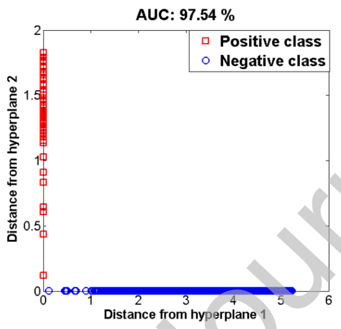


UTSVM

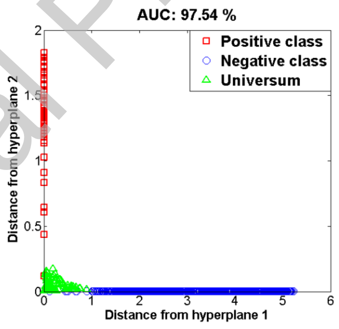


Proposed RUTSVM-CIL

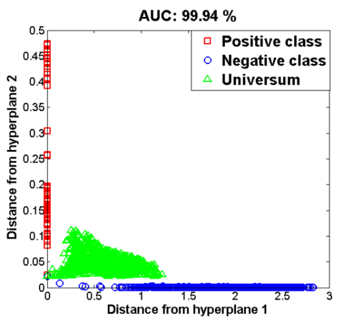
(a)



TWSVM

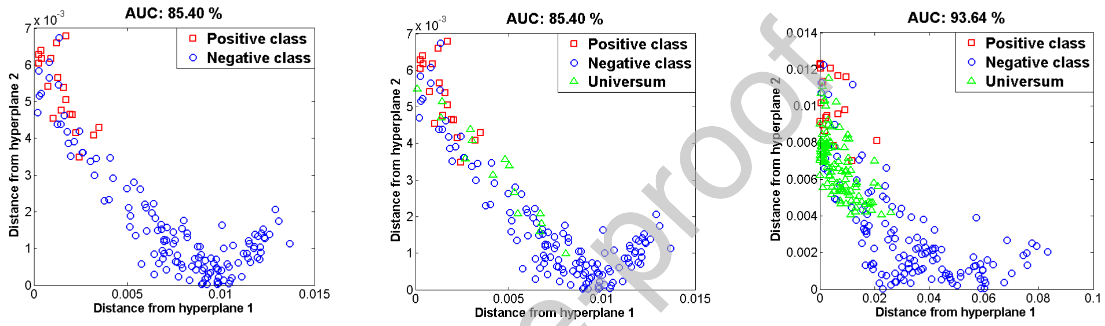


UTSVM



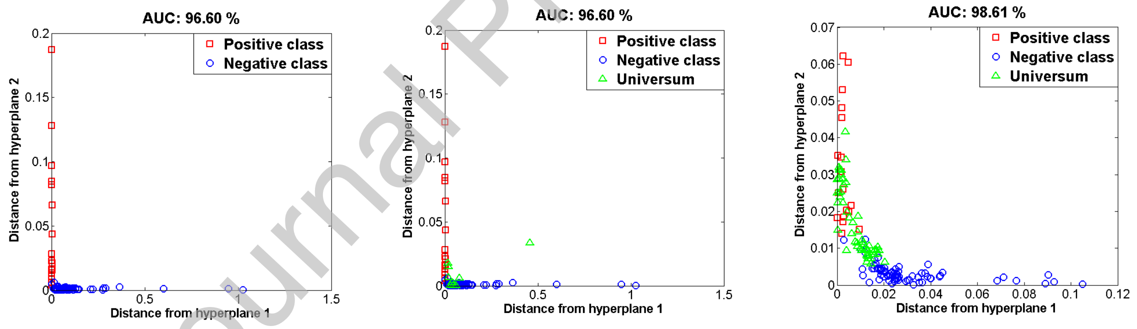
Proposed RUTSVM-CIL

(b)



TWSVM UTSVM Proposed RUTSVM-CIL

(c)



TWSVM UTSVM Proposed RUTSVM-CIL

(d)

Figure 9: Plot of distance of data points from the classifying hyperplanes for (a) Ecoli-0-4-6_vs_5, (b) Shuttle-c0-vs-c4, (c) Ecoli3 and (d) New_thyroid2.

takes lesser time while incorporating the universum data. It is visible in Table 5 that the computation time of the algorithms like SVM-RUS and TWSVM-RUS is very less as compared to other algorithms. This is due to the undersampling of data points. The computation time of FTSVM is more than TWSVM due to the calculation of fuzzy membership in FTSVM. In case of TWSVM-SMOTE, the training time is more than most of the other algorithms due to oversampling of data points. For EFSVM, the computation time is the highest, since it finds the solution of a large QPP.

5.4.3 Statistical tests

We apply the Friedman test [42] to check the statistical difference for the 9 algorithms on 28 class imbalanced datasets. First, we assume the null hypothesis as there is no difference between the methods. The F_F value for Friedman statistic is calculated using average ranks from Table 5 as

$$\chi_F^2 = \frac{12\,N}{t\times(t+1)}\left[\sum_{j=1}^{k} S_j^2 - \frac{t(t+1)^2}{4}\right],$$

where S_j is the average rank of each of the t methods for N number of datasets,

$$\chi_F^2 = \frac{12\times 28}{9\times(9+1)}\left[(4.7857^2+4.9821^2+4.6429^2+4.625^2+5.5536^2+7.2321^2+5.1429^2+5.375^2+2.6607^2)-\frac{9\times(9+1)^2}{4}\right]$$
$$\cong 41.9496$$

$$F_F = \frac{(28-1)\times 41.9496}{28\times(9-1)-41.9496} \cong 6.2216,$$

where for 9 methods and 28 datasets, the F-distribution has $(9-1,(9-1)\times(28-1))=(8,216)$ degrees of freedom. For the level of significance at $\alpha=0.05$, the critical value of $F(8,216)$ is 1.9814. Since, the value of $F_F = 6.2216 > 1.9814$, so, we reject the null hypothesis.

The Nemenyi post-hoc test [42] is carried out to verify the pair-wise difference between the proposed and existing algorithms. The critical difference is calculated as

$$CD = q_\alpha\sqrt{\frac{t\times(t+1)}{6\times N}},$$

where q_α is the critical value, CD is critical difference for t methods using N datasets.

If the value of CD at $p=0.10$ level of significance differs by at least $2.855\sqrt{\frac{9\times(9+1)}{6\times 28}} \approx 2.0896$, then there is significant difference between the methods. The results of statistical analysis are shown in Table 7.

Statistical difference	EFSVM	SVM-RUS	TWSVM	TWSVM-RUS	SMOTE	MMTSSVM	FTSVM	UTSVM
Proposed RUTSVM-CIL	Yes	Yes	No	No	Yes	Yes	Yes	Yes

Table 7: Statistical difference between RUTSVM-CIL and existing algorithms for classification on real world imbalanced datasets.

5.5 Large scale imbalanced datasets

In this subsection, we present the experimental results on large sized datasets. The proposed RUTSVM-CIL is compared with TWSVM, FTSVM, and UTSVM on NDC [40] datasets using Gaussian kernel. Table 8 shows the performance comparison on large scale imbalanced datasets, where the training time is shown in seconds. One can observe from Table 8 that the RUTSVM-CIL is performing better than the compared algorithms in most of the datasets. Moreover, the training time of RUTSVM-CIL is very less in comparison to other algorithms. This is due to the reduced kernel in our RUTSVM-CIL. For the dataset NDC-7 all the algorithms except RUTSVM-CIL failed to run due to limitation of system memory. This shows that the proposed RUTSVM-CIL is applicable on real world applications involving large scale class imbalanced data.

Moreover, the existing algorithm UTSVM is having the highest training time due to inclusion of universum data. So, the universum based algorithms are not feasible on large scale datasets. On the other hand, our RUTSVM-CIL includes the universum samples with lesser training time than all the compared algorithms.

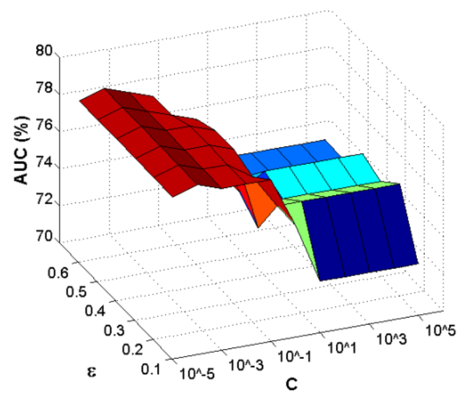
Dataset (Train size, Test size)	IR (All samples)	TWSVM AUC (%) Time (s)	FTSVM AUC (%) Time (s)	UTSVM AUC (%) Time (s)	Proposed RUTSVM-CIL AUC (%) Time (s)
NDC-1 (3336×10, 335×10)	5.67	96.1781 13.0017	95.4849 14.3013	96.3524 15.5044	**99.3031** 7.0055
NDC-2 (5008×10, 502×10)	5.68	**97.8261** 31.9098	96.0304 33.0532	96.8705 35.6478	96.755 22.2657
NDC-3 (7100×10, 712×10)	6.1	97.9305 70.6731	94.35 70.9368	96.731 74.6066	**99.0939** 42.1088
NDC-4 (13317×10, 1333×10)	5.66	98.2424 321.49	94.5994 300.514	98.9887 349.715	**99.3095** 204.397
NDC-5 (19724×10, 1974×10)	5.58	99.1583 861.113	96.67 864.887	99.2606 981.933	**99.3452** 530.71
NDC-6 (29287×10, 2930×10)	4.86	99.1033 2271.82	97.9502 2145.12	**99.2811** 2680.11	99.0635 1302.92
NDC-7 (41789×10, 4180×10)	4.97	*	*	*	99.1485 3752.11

Table 8: Comparison of proposed RUTSVM-CIL on AUC and training time with existing algorithms for classification on large scale imbalanced datasets.

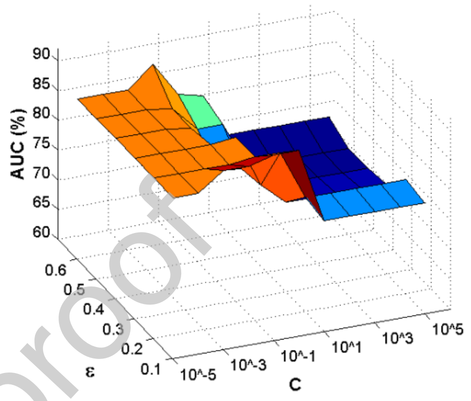
5.6 Insensitivity analysis

In Fig. 10, the insensitivity performance of the proposed RUTSVM-CIL is presented for the penalty parameter C, tolerance value ε, and kernel parameter μ. The analysis is shown for non-linear RUTSVM-CIL on the datasets Ecoli-0-1_vs_2-3-5, Ecoli-0-4-6_vs_5, Yeast1 and Yeast2vs8. Insensitivity for the parameters C and ε is shown for the datasets Ecoli-0-1_vs_2-3-5 and Ecoli-0-4-6_vs_5 in Fig. 10 (a) & (b) respectively. The value of μ is set as the optimal value obtained after cross validation. It is evident that the proposed RUTSVM-CIL gives better generalization performance for lesser values of C, and the parameter ε do not have much effect on the AUC.

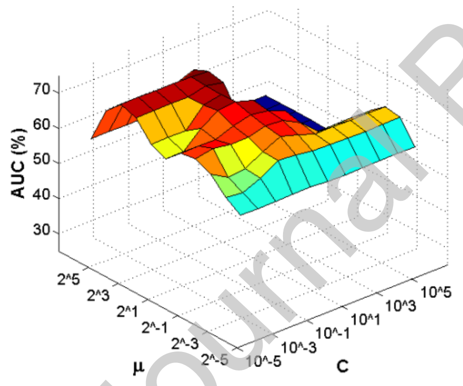
In case of insensitivity w.r.t. parameters C and μ, the value of ε is set as the optimal value obtained after cross validation. It is observable from Fig. 10. (c) & (d) that the proposed RUTSVM-CIL gives high accuracy for higher values of μ and lower values of C. This justifies the selection of the set of parameters for training of the proposed RUTSVM-CIL on class imbalanced data.



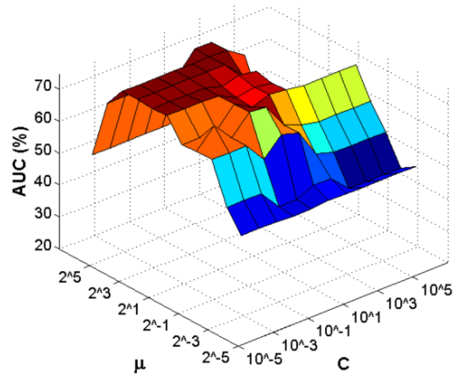
(a) Ecoli-0-1_vs_2-3-5



(b) Ecoli-0-4-6_vs_5



(c) Yeast1



(d) Yeast2vs8

Figure 10: Insensitivity performance of proposed RUTSVM-CIL for classification of real world imbalance datasets to the user specified parameters using Gaussian kernel.

6. Conclusions & future directions

In this paper, we proposed a novel computationally efficient model i.e., reduced universum twin support vector machine for class imbalance learning. The proposed model incorporates prior information from the universum data, and creates a balance situation for the classification. The reduced kernel based approach leads to a computationally efficient model of universum based SVM. This removes the overhead of higher computation cost of universum based algorithms. The memory requirement for executing the proposed algorithm is also very less, which makes it suitable for large scale imbalanced datasets. The approach of combining undersampling with oversampling using universum data is found to be helpful in classification of class imbalance datasets. Our model has shown good generalization performance with less training time on several synthetic and real world datasets. Moreover, RUTSVM-CIL shows high efficiency for large scale datasets with better classification accuracy. However, due to the use of undersampling and rectangular kernel, the proposed RUTSVM-CIL gives lesser accuracy for high imbalance ratio, but with lesser computation cost.

In future, the accuracy of the proposed RUTSVM-CIL can be improved by proper selection of universum. The proper selection of universum is also a field of research, and depends on the type of application. The proposed approach can be used with proper selection of universum from the imbalanced data itself. Further, different types of undersampling and oversampling techniques can be used for our RUTSVM-CIL to improve the performance. Multiclass classification of data can also be performed by modifying the proposed RUTSVM-CIL. Since, RUTSVM-CIL involves sampling of data with reduced kernel, it can be useful on applications involving undersampling or oversampling of data. The proposed model can be applied on classification problems involving large scale imbalanced datasets. Also, RUTSVM-CIL can be applied for the diagnosis of various diseases that involve high imbalance in samples of healthy and disease cases.

Acknowledgements

The authors would like to thank the Editor and the anonymous reviewers whose constructive comments and feedback have helped us to improve the content of the whole paper.

This work was supported by Science and Engineering Research Board (SERB) as Early Career Research Award grant no. ECR/2017/000053 and Department of Science and Technology as Ramanujan fellowship grant no. SB/S2/RJN-001/2016. We gratefully acknowledge Indian Institute of Technology Indore for providing facilities and support. Sincere thanks to Indian Institute of Technology Indore for providing Teaching Assistantship (TA) fellowship to Mr. Bharat Richhariya.

References

[1]. C. Cortes, V. Vapnik, Support vector networks, Machine learning, 20 (3) (1995) 273-297.

[2]. B. Richhariya, M. Tanveer, EEG signal classification using universum support vector machine, Expert Systems with Applications, 106 (2018) 169-182.

[3]. D. Tomar, S. Agarwal, Hybrid feature selection based weighted least squares twin support vector machine approach for diagnosing breast cancer, hepatitis, and diabetes, Advances in Artificial Neural Systems (2015) 1.

[4]. X. Zhang, D. Jiang, T. Han, N. Wang, W. Yang, Y. Yang, Rotating machinery fault diagnosis for imbalanced data based on fast clustering algorithm and support vector machine, Journal of Sensors (2017).

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

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

[7]. X. Peng, A ν-twin support vector machine (ν-TSVM) classifier and its geometric algorithms, Information Sciences, 180 (20) (2010) 3863-3875.

[8]. C. Wang, Q. Ye, P. Luo, N. Ye, L. Fu, Robust capped L1-norm twin support vector machine, Neural Networks, 114 (2019) 47-59.

[9]. H. Yan, Q. Ye, T.A. Zhang, D.J. Yu, X. Yuan, Y. Xu, L. Fu, Least squares twin bounded support vector machines based on L1-norm distance metric for classification, Pattern Recognition, 74 (2018) 434-447.

[10]. H. Zhao, Q. Ye, M.A. Naiem, L. Fu, Robust $L_{2,1}$ -norm distance enhanced multi-weight vector projection support vector machine, IEEE Access, 7 (2019) 3275-3286.

[11]. R. Rezvani-KhorashadiZadeh, M. Reza, WS-TWSVM: Weighted structural twin support vector machine by local and global information. In 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE), IEEE (2015) 170-175, October.

[12]. S. Sun, X. Xie, C. Dong, Multiview learning with generalized eigenvalue proximal support vector machines, IEEE Transactions on Cybernetics, 49 (2) (2018) 688-697.

[13]. X. Xie, Regularized multi-view least squares twin support vector machines, Applied Intelligence, 48 (9) (2018) 3108-3115.

[14]. H. Cevikalp, Best fitting hyperplanes for classification, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (6) (2017) 1076-1088.

[15]. S. Ding, X. Zhang, Y. An, Y. Xue, Weighted linear loss multiple birth support vector machine based on information granulation for multi-class classification. Pattern Recognition, 67 (2017) 32-46.

[16]. J. Weston, R. Collobert, F. Sinz, L. Bottou, V. Vapnik, Inference with the universum. In Proceedings of the 23rd International Conference on Machine learning, ACM (2006) 1009-1016, June.

[17]. O. Chapelle, A. Agarwal, F.H. Sinz, B. Schölkopf, An analysis of inference with the universum, In Advances in Neural Information Processing Systems (2008) 1369-1376.

[18]. Z. Qi, Y. Tian, Y. Shi, Twin support vector machine with universum data, Neural Networks, 36 (2012) 112-119.

[19]. Y. Xu, M. Chen, G. Li, Least squares twin support vector machine with universum data for classification, International Journal of Systems Science, 47 (15) (2016) 3637-3645.

[20]. R. Batuwita, V. Palade, FSVM-CIL: Fuzzy support vector Machines for class imbalance learning, IEEE Transactions on Fuzzy Systems, 18 (3) (2010) 558-571.

[21]. B.X. Wang, N. Japkowicz, Boosting support vector machines for imbalanced data sets, Knowledge and Information Systems, 25 (1) (2010) 1-20.

[22]. N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research, 16 (2002) 321-357.

[23]. Y. Tang, Y.Q. Zhang, N.V. Chawla, S. Krasser, SVMs modeling for highly imbalanced classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39 (1) (2009) 281-288.

[24]. X.Y. Liu, J. Wu, Z.H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39 (2) (2009) 539-550.

[25]. D.J. Yu, J. Hu, Z.M. Tang, H.B. Shen, J. Yang, J.Y. Yang, Improving protein-ATP binding residues prediction by boosting SVMs with random under-sampling, Neurocomputing, 104 (2013) 180-190.

[26]. K. Li, H. Ma, A fuzzy twin support vector machine algorithm, International Journal of Application or Innovation in Engineering and Management (IJAIEM), 2 (3) (2013) 459-465.

[27]. Q. Fan, Z. Wang, D. Li, D. Gao, H. Zha, Entropy-based fuzzy support vector machine for imbalanced datasets, Knowledge-Based Systems, 115 (2017) 87-99.

[28]. B. Richhariya, M. Tanveer, A robust fuzzy least squares twin support vector machine for class imbalance learning, Applied Soft Computing, 71 (2018) 418-432.

[29]. M. Tanveer, A. Sharma, P.N. Suganthan, General twin support vector machine with pinball loss function, Information Sciences, 494 (2019) 311-327.

[30]. Y. Xu, Maximum margin of twin spheres support vector machine for imbalanced data classification, IEEE Transactions on Cybernetics, 47 (6) (2017) 1540-1550.

[31]. Y. Xu, Z. Yang, Y. Zhang, X. Pan, L. Wang, A maximum margin and minimum volume hyper-spheres machine with pinball loss for imbalanced data classification, Knowledge-Based Systems, 95 (2016) 75-85.

[32]. Y. Xu, Y. Zhang, J. Zhao, Z. Yang, X. Pan, KNN-based maximum margin and minimum volume hyper-sphere machine for imbalanced data classification, International Journal of Machine Learning and Cybernetics, (2017) 1-12.

[33]. Y.J. Lee, O.L. Mangasarian, RSVM: Reduced support vector machines. In Proceedings of the 2001 SIAM International Conference on Data Mining (April, 2001) pp. 1-17, Society for Industrial and Applied Mathematics.

[34]. M. Singh, J. Chadha, P. Ahuja, Jayadeva, S. Chandra, Reduced twin support vector regression, Neurocomputing, 74 (9) (2011) 1474-1477.

[35]. Y.J. Lee, S.Y. Huang, Reduced support vector machines: A statistical theory, IEEE Transactions on Neural Networks, 18 (1) (2007) 1-13.

[36]. A. Karami, An anomaly-based intrusion detection system in presence of benign outliers with visualization capabilities, Expert Systems with Applications, 108 (2018) 36-60.

[37]. Y.H. Shao, C.H. Zhang, X.B. Wang, N.Y. Deng, Improvements on twin support vector machines, IEEE Transactions on Neural Networks, 22 (6) (2011) 962-968.

[38]. J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic & Soft Computing, 17 (2011).

[39]. P.M. Murphy, D.W. Aha, UCI repository of machine learning databases, University of California, Irvine. http://www.ics.uci.edu/~mlearn (1992).

[40]. D.R. Musicant, NDC: Normally distributed clustered datasets. Computer Sciences Department, University of Wisconsin, Madison (1998).

[41]. A. Luque, A. Carrasco, A. Martín, A. de las Heras, The impact of class imbalance in classification performance metrics based on the binary confusion matrix. Pattern Recognition, 91 (2019) 216-231.

[42]. J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research, 7 (Jan) (2006) 1-30.

B. Richhariya: Mr. Bharat Richhariya completed his B. Tech. degree in Computer Science & Engineering from National Institute of Technology Mizoram, India in the year 2014. He completed his M.Tech. in Computer Science & Engineering from National Institute of Technology Arunachal Pradesh, India in the year 2017. He is currently pursuing his Ph.D. in Mathematics at Indian Institute of Technology Indore, India. His research interests include machine learning, biomedical signal processing and computer vision.

M. Tanveer: Dr. Tanveer is Assistant Professor and Ramanujan Fellow at the Discipline of Mathematics of the Indian Institute of Technology, Indore. Prior to that, he spent one year as a Postdoctoral Research Fellow at the Rolls-Royce@NTU Corporate Lab of the Nanyang Technological University, Singapore. His research interests include support vector machines, optimization, applications to Alzheimer's disease and dementias, biomedical signal processing, and fixed point theory and applications. He has published over 25 referred journal papers of international repute. He is the recipient of the 2016 DST-Ramanujan Fellowship in Mathematical Sciences and 2017 SERB-Early Career Research Award in Engineering Sciences which are the prestigious awards of INDIA at early career level. He is a member of the editorial review board of Applied Intelligence, Springer (International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies) and lead Guest Editor of ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM).

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.