Development of robust support vector machine algorithms with biomedical applications

Ph.D. Thesis

By

Bharat Richhariya

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY INDORE

APRIL 2021

Development of robust support vector machine algorithms with biomedical applications

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

in partial fulfillment of the requirements for

the award of the degree

of

DOCTOR OF PHILOSOPHY

By

Bharat Richhariya

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY INDORE

APRIL 2021

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Development of robust support vector machine algorithms with biomedical applications** in the partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted in the **Department of Mathematics, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the time period from December 2017 to April 2021 under the supervision of **Dr. M. Tanveer**, Associate Professor, Indian Institute of Technology Indore, Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Bharat Richhariya
13-04-2021

Signature of the Student with Date

(Bharat Richhariya)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

April 13, 2021

Signature of Thesis Supervisor with Date

(Dr. M. Tanveer)

Bharat Richhariya has successfully given his Ph.D. Oral Examination held on **30 July 2021**.

July 30, 2021

Signature of Thesis Supervisor with Date

(Dr. M. Tanveer)

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the persons who in some way helped me in completing my PhD work. Firstly, I would like to thank my Thesis supervisor **Dr. M. Tanveer**, for the continuous guidance and support. He provided the motivation and support, which helped me in completing my research work. His expertise in the research field helped me a lot in tackling the problems arising in the various stages of research. This lead to the timely completion of my PhD thesis.

I am also thankful to **Dr. Niraj Kumar Shukla** and **Dr. Aruna Tiwari**, my research progress committee members for giving valuable comments, which lead to improvements in my research work. I am also thankful to the Head, Department of Mathematics for his consistent support and encouragement.

My sincere gratitude and respect to Director, Indian Institute of Technology Indore for providing the required research facilities for my work.

I would acknowledge Indian Institute of Technology Indore for providing me institute fellowship for my PhD programme.

I would like to thank my lab mates for the healthy scientific discussions and exchange of ideas, especially Ashraf, Ashwani, Anshul, Riyaz, Mudasir, Avijit, and Shilpa. I am also thankful to Chandan, and Pratik for the helpful discussions on machine learning. Moreover, I would thank my colleagues from my department for discussions on mathematical topics especially, Shreyas, Shubham, Sudipta, Prince, Vibhuti, Vineeta, and Arstu.

I am also grateful to the administrative and other staff of the institute for their support and assistance, wherever required during my PhD.

I am thankful to my parents for their constant support and encouragement, which lead to the smooth completion of my PhD.

Finally, I would like to express my PhD journey with the following line:

"The path not taken is the path to discovery."

Bharat Richhariya

Bharat Richhariya

To my family and friends

Abstract

The work presented in this thesis comprises robust and efficient machine learning (ML) models based on novel optimization approaches. The ML technique investigated very thoroughly in this work is support vector machine (SVM). SVM is a widely used supervised learning algorithm for classification as well as regression problems. It uses a kernel based approach for efficiently classifying the data. Since SVM based algorithms have been extensively used for classifying biomedical data [1, 2], we applied most of the proposed SVM models to biomedical applications.

Our survey identified some key problems in training SVMs, viz. (which are) class imbalance problem, data with noise, no knowledge about data distribution, unlabelled data, and proper feature selection. Also, these problems often occur with biomedical data as well. To resolve these issues, we proposed novel SVM based algorithms involving universum data and fuzzy logic. We presented the applications of these models for healthcare, such as automated diagnosis of diseases like brain disorders. Moreover, we focused on the issue of proper integration of machine learning algorithms with specific applications. We also performed a review of the various machine learning techniques used in detecting brain disorders. This resulted in the detection of key problems related to machine learning usage in the biomedical domain. One of the key issues is identifying features containing possible locations of brain regions responsible for the disease. To resolve this, we proposed a novel feature selection technique based on universum SVM in this thesis.

To deal with the problem of class imbalance, we proposed two novel algorithms for classification. One of the algorithm is termed as a robust fuzzy least squares twin support vector machine for class imbalance learning (RFLSTSVM-CIL). The RFLSTSVM-CIL algorithm removes the class imbalance problem using a fuzzy logic based approach, which in turn helps to deal with noisy data as well. The second algorithm is proposed using a different approach involving universum data to use prior information about data distribution for class imbalance scenarios. This algorithm is termed as a reduced universum twin support vector machine for class imbalance

learning (RUTSVM-CIL).

We utilized universum learning for neurological disorders in this work. For epilepsy, we used electroencephalogram (EEG) recordings to propose a universum SVM based seizure detection technique. Moreover, for feature selection, we proposed a universum based feature elimination algorithm, termed as universum support vector machine based recursive feature elimination (USVM-RFE). We applied the proposed USVM-RFE on Alzheimer's disease (AD) using magnetic resonance imaging (MRI) data for classification.

However, to deal with noisy datasets in universum learning, three fuzzy logic based universum algorithms are proposed in this thesis as: A fuzzy universum support vector machine (FUSVM), a fuzzy universum twin support vector machine (FUTSVM) based on information entropy, and a fuzzy universum least squares twin support vector machine (FULSTSVM). However, universum learning incurs additional computation time. Therefore, we presented some efficient universum based SVM algorithms. We proposed an efficient angle based universum least squares twin support vector machine (AULSTSVM) for pattern classification. AULSTSVM uses an angle based approach for universum learning. Also, two novel variants of universum based twin SVM algorithms are proposed as: Improved universum twin support vector machine (IUTSVM), and universum least squares twin parametric-margin support vector machine (ULSTPMSVM). Most of the above mentioned algorithms are applied on Alzheimer's disease data for detection of disease.

To explore the domain of unsupervised learning, we presented an SVM based algorithm, termed as least squares projection twin support vector clustering (LSPTSVC), and applied on AD data. All the models proposed in thesis are compared with baseline algorithms to justify the advantages. The results of numerical experiments are compared using statistical significance tests.

Keywords: Support vector machine, class imbalance, universum data, fuzzy membership, Alzheimer's disease, epilepsy, MRI, EEG, twin support vector machine, clustering.

List of Publications

A. Published

A1. In Refereed Journals

1. **B. Richhariya**, M. Tanveer. A robust fuzzy least squares twin support vector machine for class imbalance learning. *Applied Soft Computing*, Elsevier, 71: 418-432, 2018, DOI: https://doi.org/10.1016/j.asoc.2018.07.003.
   [**SCI Indexed Impact Factor: 6.725**]

2. **B. Richhariya**, M. Tanveer. A reduced universum twin support vector machine for class imbalance learning. *Pattern Recognition*, Elsevier, 102:107150, 2020, DOI: https://doi.org/10.1016/j.patcog.2019.107150.
   [**SCI Indexed Impact Factor: 7.740**]

3. **B. Richhariya**, M. Tanveer. EEG signal classification using universum support vector machine. *Expert Systems with Applications*, Elsevier, 106:169-182, 2018, DOI: https://doi.org/10.1016/j.eswa.2018.03.053.
   [**SCI Indexed Impact Factor: 6.954**]

4. **B. Richhariya**, M. Tanveer, A.H. Rashid, Alzheimer's Disease Neuroimaging Initiative. Diagnosis of Alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE). *Biomedical Signal Processing and Control*, Elsevier, 59:101903, 2020, DOI: https://doi.org/10.1016/j.bspc.2020.101903.
   [**SCI Indexed Impact Factor: 3.880**]

5. **B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. Least squares projection twin support vector clustering (LSPTSVC). *Information Sciences*, Elsevier, 533:1-23, 2020, DOI: https://doi.org/10.1016/j.ins.2020.05.001.
   [**SCI Indexed Impact Factor: 6.795**]

6. **B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. A fuzzy universum least squares twin support vector machine (FULSTSVM). *Neural Computing and Applications*, Springer, 2021, DOI: https://doi.org/10.1007/s00521-021-05721-4.
   [**SCI Indexed Impact Factor: 5.606**]

7. **B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. An efficient angle based universum least squares twin support vector machine for pattern classification. *ACM Transactions on Internet Technology (TOIT)*, ACM, 2021, DOI: https://doi.org/10.1145/3387131.
   [**SCI Indexed Impact Factor: 3.135**]

8. M. Tanveer, **B. Richhariya**, R.U. Khan, A.H. Rashid, P. Khanna, M. Prasad, C.T. Lin. Machine learning techniques for the diagnosis of Alzheimer's disease: A review. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, ACM, 16(1s):1-35, 2020, DOI: https://doi.org/10.1145/3344998.
   [**SCI Indexed Impact Factor: 3.144**]

A2. In Refereed Conferences

1. **B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. Universum least squares twin parametric-margin support vector machine. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1-8. IEEE, 2020, DOI: https://doi.org/10.1109/IJCNN48605.2020.9206865.
   [**Scopus Indexed, Core rank: A**]

2. **B. Richhariya**, M. Tanveer. A fuzzy universum support vector machine based on information entropy. In M. Tanveer and Ram Bilas Pachori, editors, *Machine Intelligence and Signal Analysis*, volume 748, pages 569–582. Springer Singapore, 2019, DOI: https://doi.org/10.1007/978-981-13-0923-6_49.
   [**Scopus Indexed**]

3. **B. Richhariya**, A. Sharma, M. Tanveer. Improved universum twin support vector machine. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2045-2052. IEEE, 2018, DOI: https://doi.org/10.1109/SSCI.2018.8628671.
[**Scopus Indexed**]

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

SVM Support vector machine

SRM Structural risk minimization

TWSVM Twin support vector machine

LSTSVM Least squares twin support vector machine

USVM Universum support vector machine

RFLSTSVM-CIL Robust fuzzy least squares twin support vector machine for class imbalance learning

RUTSVM-CIL Reduced universum twin support vector machine for class imbalance learning

RFE Recursive feature elimination

MRI Magnetic resonance imaging

HC Healthy control

MCI Mild cognitive impairment

AD Alzheimer's disease

EEG Electroencephalogram

FUSVM Fuzzy universum support vector machine

FUTSVM Fuzzy universum twin support vector machine

FULSTSVM Fuzzy universum least squares twin support vector machine

AULSTSVM Angle based universum least squares twin support vector machine for pattern classification

IUTSVM Improved universum twin support vector machine

ULSTPMSVM Universum least squares twin parametric-margin support vector machine

TWSVC Twin support vector clustering

LSPTSVC Least squares projection twin support vector clustering

Chapter 1

Introduction

In the past few decades, the data generation has increased substantially, creating a need for development of robust machine learning techniques. Various improvements in algorithms have lead to accurate and efficient learning techniques, making remarkable changes in our daily lives. Many of the widely used machine learning algorithms belong to a class known as supervised learning. Support vector machines proposed by Vapnik and co-workers [4–6] have turned out to be a successful algorithm for classification problems. SVM is based on the maximal margin principle, which leads to an optimal hyperplane for classification of data. SVM and its variants have been applied to solve various real world problems ranging from classification of neurological disorders [7] to applications such as fingerprint [8] and facial expression recognition [9]. In order to improve the generalization performance of SVM, Weston et al. [10] came up with the idea of universum data. The universum data are unlabelled, and lie in between the binary classes. Other than universum SVM, many other variants of SVM have been developed in the past to improve its performance.

In this thesis, we present novel variants of SVM, and apply them on various biomedical applications. The biomedical data included in this work is primarily related to neurological disorders, namely Alzheimer's disease, and epilepsy. However, one application is also presented on breast cancer data. Like any other classification problem involving high dimensional data, the neurological disorders are hard to detect without the use of artificial intelligence. Currently, 50 million people are affected worldwide

by dementia, and is expected to be 152 million by the year 2050 [11]. Alzheimer's disease (AD) is a progressive neurodegenerative disease, primarily affecting the elderly population. It is also a leading cause of dementia. Diagnosis of AD is a formidable task that requires a lot of expertise, and a thorough examination of patient data. Moreover, for epilepsy detection, electroencephalogram (EEG) signal classification is a major challenge in the field of machine learning and signal processing. EEG is a widely used non-invasive technique for the detection of various types of brain disorders such as epileptic seizures and sleep disorders. In epilepsy, the extent of disease ranges from partial to generalized seizures which are reflected in their respective EEG.

Various other challenges need to be addressed with data, such as unlabelled and noisy data. Moreover, robust algorithms need to be developed which are insensitive to small variations in the data distribution. Some of these challenges are addressed in this thesis with a focus on biomedical applications.

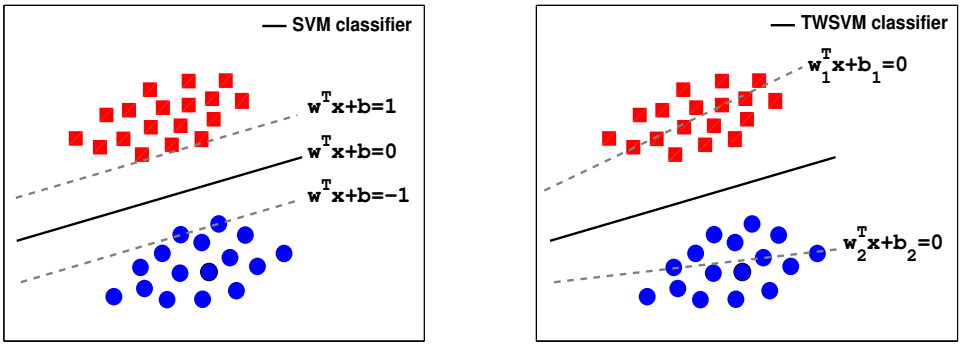
1.1 Background

Support vector machine (SVM) [5] is a widely used technique for classification [2,12] and regression problems [13, 14]. Based on the structural risk minimization principle (SRM), SVM gives very good generalization performance. SVM uses the maximal margin principle to classify the data points as shown in Fig. 1.1(a). After solving a convex optimization problem, the decision function of SVM is written as,

$$f(x) = sign\ (w^T x + b), \tag{1.1}$$

where w^T is transpose of weight vector w, and b is the bias.

To classify non-linearly separable data, kernel functions [15] have been used to transform the data to higher dimensions. Moreover, various variants of SVM have been proposed to increase its performance with respect to (w.r.t.) generalization ability and training time [16]. A computationally efficient variant of SVM is proposed by Jayadeva et al. [12], known as twin support vector machine (TWSVM). The TWSVM algorithm



(a) SVM (b) TWSVM

Figure 1.1: Types of SVM classifiers.

generates twin hyperplanes to classify the data as shown in Fig. 1.1(b). To improve the computation cost of TWSVM, least squares based algorithms are proposed, such as least squares SVM (LSSVM) [17], and least squares twin support vector machine (LSTSVM) [18].

In 2006, Weston et al. [10] proposed a novel universum based support vector machine (USVM) by incorporating universum data in the formulation of SVM. The universum data consists of additional data points not belonging to any of the binary classes. Universum data gives prior information about data distribution to the classifier. Cherkassky et al. [19] stated the practical conditions on the effectiveness of universum such as selection of parameters. Due to the higher generalization performance, universum based algorithms have been used in various applications such as classification of EEG signals [20], gender [21], and investor sentiments [22].

The universum data is used to align the classifier with the data distribution. As shown in Fig. 1.2, the classifier generated by USVM is better aligned to classify the data points. This helps in the classification of testing data. Without this knowledge of the data distribution, the SVM classifier only tries to maximize the margin, which results in reduced generalization performance of the model.

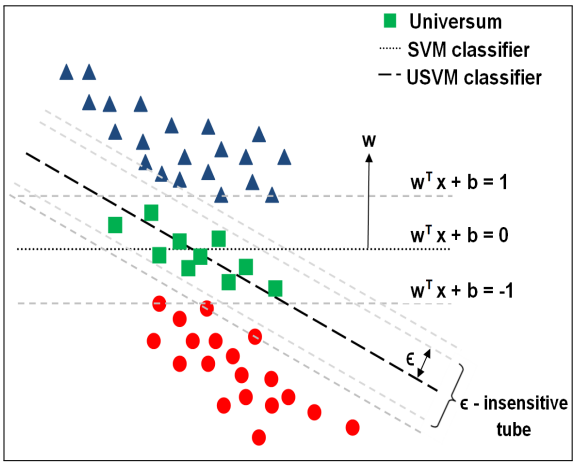


Figure 1.2: Universum data.

1.2 Motivation

Machine learning (ML) techniques are found to be very useful in various real world problems [2] in the last decade. Among the various ML techniques, we chose to work on SVM due to its significant prevalence in the literature for ML applications [1]. For understanding the status of state of the art, we surveyed a number of papers using SVM for classification. We observed that SVM based models have been widely used for various types of data. We also found various problems involved with SVM for classification tasks such as **class imbalance, noisy data, lack of prior information about data distribution, and unlabelled data**. This motivated us to develop novel SVM algorithms to remove these drawbacks.

Moreover, we found that SVM based models have been extensively used for biomedical data, such as EEG [23] and MRI [2]. It is due to transparency in the SVM model for the relation between features and prediction values. Therefore, we proposed novel variants of SVM and applied on brain disorder datasets. Also, it is observed that the classification of two classes can be improved by using a third class, known as universum learning [10]. We found that universum based SVM (USVM) [10] is not effectively utilized in the past for classification, especially for biomedical datasets. This motivates us to develop novel USVM models for classification tasks, such as seizure detection for epilepsy.

However, we found that in the case of Alzheimer's disease data, better classification requires identifying brain regions responsible for the disease [24]. Therefore, we proposed a feature elimination method based on USVM. Moreover, the availability of a lot of unlabelled data in today's world motivates us to explore the domain of unsupervised learning as well. This leads us to develop a SVM based clustering algorithm with biomedical applications.

1.3 Objectives

The objectives of this thesis are as follows:

(i). To develop SVM based classifiers for class imbalanced data.
(ii). To present a review on the works based on SVM, especially those involving biomedical data.
(iii). To develop classification techniques for brain disorders using universum learning.
(iv). To formulate noise insensitive universum SVM based classifiers using fuzzy logic.
(v). To propose efficient universum based twin SVM algorithms.
(vi). To propose a novel SVM based unsupervised learning technique.

1.4 Thesis contributions

In this section, we give a brief overview on the contributions of our work. The proposed algorithms are abbreviated with bold font. The contributions are as follows:

I. Twin support vector machine for class imbalance learning

Based on our review, it is found that noisy class imbalanced data pose a major challenge in various applications. To resolve this problem, we propose a robust fuzzy least squares twin support vector machine for class imbalance learning, termed as **RFLSTSVM-CIL** [25]. In order to reduce the effect of outliers, we propose a novel fuzzy membership function specifically for class imbalance problems.

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, a reduced universum twin support vector machine for class imbalance learning (**RUTSVM-CIL**) [26] is proposed in this thesis. 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 class imbalance. The universum data gives prior information about the distribution of data. To reduce the computation time of our universum based algorithm, we use a small sized rectangular kernel matrix.

II. Review on machine learning techniques for Alzheimer's disease

To develop efficient learning techniques, a better understanding of the existing work is needed. Therefore, we reviewed various papers from 2005-2019 on the works involving feature extraction and machine learning techniques for Alzheimer's disease. The machine learning techniques are surveyed under three main categories: support vector machine (SVM), artificial neural network (ANN), and deep learning (DL) with ensemble methods [2]. We present a detailed review on the use of SVM based approaches for AD in this thesis with possible future directions.

III. Universum learning for neurological disorders

In our survey [2], we found that apart from class imbalance, noise poses a problem in balanced data as well. In order to develop a robust classifier, we propose universum based techniques for neurological disorders such as epilepsy. We present a novel machine learning approach based on universum support vector machine (USVM) to include prior information about data. In our approach, the universum data points are generated by selecting universum from the EEG dataset itself, which are the interictal EEG signals [20]. This removes the effect of outliers on the generation of universum data. Further, to reduce the computation time, we use our approach of universum selection with universum twin support vector machine (UTSVM) [27].

Moreover, other than classification algorithms, efficient feature extraction techniques are also needed for common biomedical data, such as MRI images. Motivated by the work on support vector machine based recursive feature elimination (SVM-RFE) [1], we propose a novel feature selection technique to incorporate prior informa-

tion about data distribution in the recursive feature elimination process. Our method is termed as universum support vector machine based recursive feature elimination (**USVM-RFE**) [28]. For application, we applied the proposed approach on AD data. The proposed method provides global information about data in the RFE process as compared to the local approach of feature selection in SVM-RFE.

IV. Fuzzy universum support vector machines

Although universum learning provides prior information about data distribution, noise is still a major cause for mis-classification. Therefore, to improve noise insensitivity for universum based algorithms, we propose fuzzy based universum SVM algorithms. First, we present a fuzzy universum support vector machine (**FUSVM**) [29] by introducing weights to the universum data points based on their information entropy. In addition, we also propose an efficient variant of this approach as fuzzy universum twin support vector machine (**FUTSVM**).

To further reduce the computation time, a least squares based model is proposed as fuzzy universum least squares twin support vector machine (**FULSTSVM**) [30]. In FULSTSVM, the membership values are used to provide weights for data samples of the classes, as well as to the universum.

V. Efficient universum twin support vector machines

Universum learning incurs better generalization performance. However, it involves a drawback of additional computation time. To improve the efficiency of universum based algorithms, we present novel universum twin support vector machines to generate two hyperplanes. First, we propose an efficient approach termed as angle based universum least squares twin support vector machine (**AULSTSVM**) [31]. This is a novel approach of incorporating universum in the formulation of least squares based twin SVM.

Moreover, a novel parametric model for universum based twin support vector machine is presented for classification problems. The proposed model is termed as universum least squares twin parametric-margin support vector machine (**ULSTPMSVM**) [32]. The solution of ULSTPMSVM involves a system of linear equations, making it

efficient in terms of training time.

In order to include the structural risk minimization (SRM) principle in the formulation of UTSVM, we propose an improved universum twin support vector machine (**IUTSVM**) [33]. Our proposed IUTSVM implicitly makes the matrices non-singular in the optimization problem by including a regularization term.

VI. Projection based twin support vector clustering

Most of the algorithms proposed in this thesis involve supervised learning. In order to learn from unlabelled data, we propose an unsupervised learning algorithm based on projection axes, termed as least squares projection twin support vector clustering (**LSPTSVC**) [34]. The proposed LSPTSVC finds projection axis for every cluster in a manner that minimizes the within class scatter, and keeps the clusters of other classes far away. Moreover, the solution of proposed LSPTSVC involves a set of linear equations leading to very less computation time.

1.5 Organization of the thesis

The works included in this thesis are divided into eight chapters. Fig. 1.3 shows a pictorial representation of the works. We have given a brief description of every chapter in the following:

Chapter 1 (Introduction)

In this chapter, we provided the introduction, and background of SVM. We explained the motivation for this work, with the contributions made in this thesis.

Chapter 2 (Literature Survey and Research Methodology)

This chapter provides a thorough review on the existing SVM based learning techniques, with applications to biomedical data. It also describes the research methodology including the performance metrics used in this thesis.

Chapter 3 (Twin support vector machine for class imbalance learning)

In this chapter, we present two novel SVM based algorithms for class imbalanced

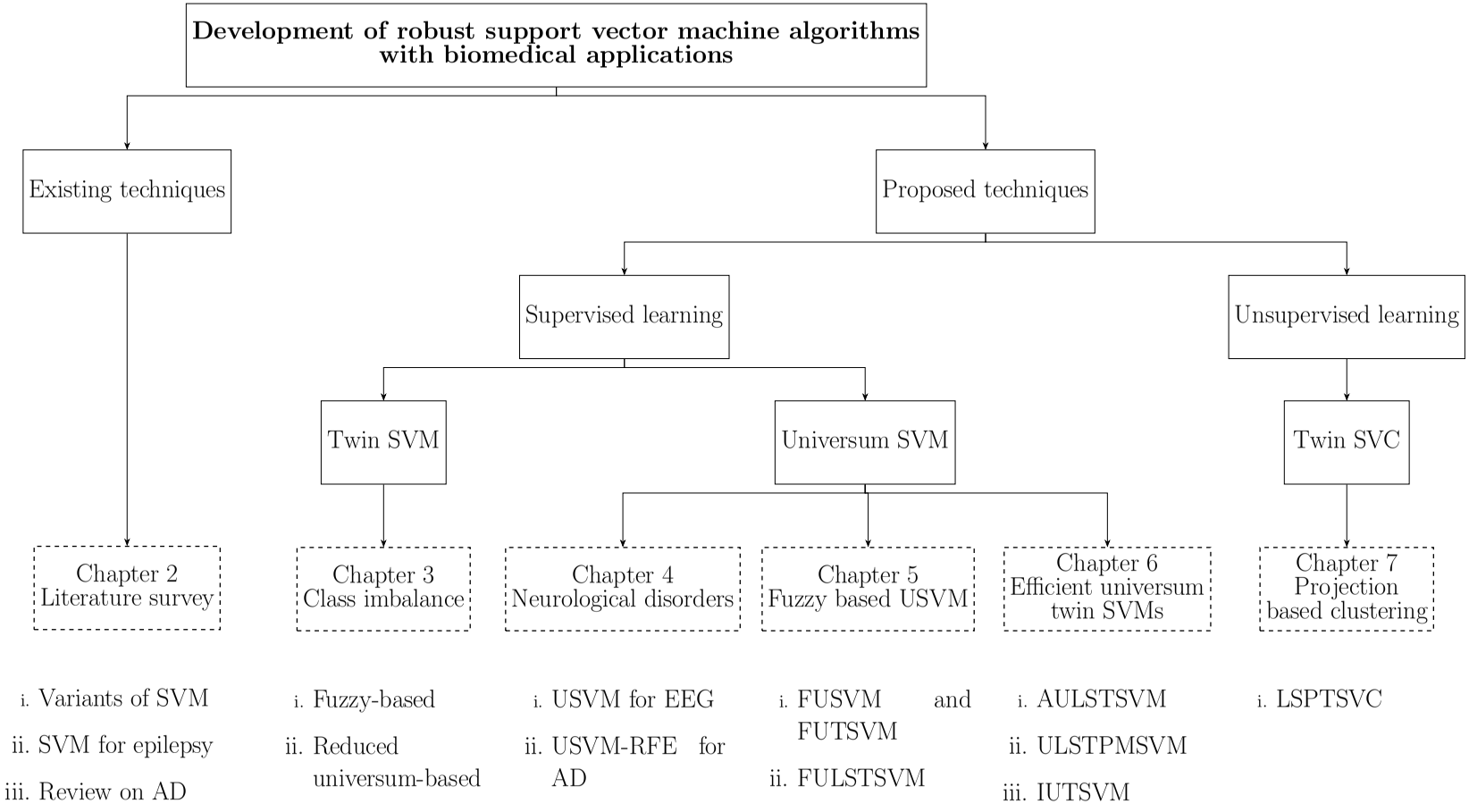


Figure 1.3: Flowchart of works presented in this thesis.

data.

Chapter 4 (Universum learning for neurological disorders)

This chapter presents a novel universum based approach for detection of epilepsy. Also, a universum based feature elimination technique is proposed for Alzheimer's disease.

Chapter 5 (Fuzzy universum support vector machines)

To remove the effect of noise, three novel universum SVM based algorithms are presented using fuzzy logic.

Chapter 6 (Efficient universum twin support vector machines)

This chapter discusses three novel universum SVM based algorithms with improved generalization and lesser computation time.

Chapter 7 (Projection based twin support vector clustering)

For unsupervised learning, a twin support vector clustering technique is proposed using projection of data points.

Chapter 8 (Conclusions and Future Work)

This chapter gives the conclusions of the thesis, with possible future directions.

Chapter 2

Literature Survey and Research Methodology

In this chapter, we present the literature on the different research problems addressed in this thesis. We have divided the literature survey into six sections, with section 2.1 introducing the formulation of SVM. Section 2.2 discusses twin SVM based techniques in reference to class imbalance problems and fuzzy functions. Section 2.3 describes the formulation of USVM with applications to neurological disorders. Moreover, a survey on machine learning for Alzheimer's disease[1] is presented in section 2.4. For unsupervised learning, the works on SVM based clustering algorithms are explained in section 2.5. Lastly, section 2.6 gives the research methodology used in this work.

Notations: The mathematical notations used in this work are as follows: All vectors x are assumed as column vectors. x^T denotes the transpose of the vector. X_1 and X_2 are matrices containing the data points belonging to class '1' and '-1' of size $p \times n$ and $q \times n$ respectively. U represents universum data points having dimension $r \times n$. Total number of data points are represented by $l = p + q$, where n is the dimension of each data point. $\|x\|$ represents the 2-norm of a vector x.

[1]M. Tanveer, **B. Richhariya**, R.U. Khan, A.H. Rashid, P. Khanna, M. Prasad, C.T. Lin. Machine learning techniques for the diagnosis of Alzheimer's disease: A review. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, ACM, 16(1s):1-35, 2020, DOI: `https://doi.org/10.1145/3344998`.
[**SCI Indexed Impact Factor: 3.144**]

In the last few decades, support vector machine (SVM) [5] has become a popular technique for classification problems [16]. SVM is based on the maximal margin principle with bounded VC dimension, leading to better generalization performance of the model. In the next section, first we present the mathematical formulation of SVM, then in further sections, we discuss about the various improvements made on the SVM formulation.

2.1 Support vector machine for classification problems

The formulation of SVM [5] in primal form is written as follows:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{l}\xi_i \\ s.t. \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \forall i = 1, 2, \ldots, l, \end{aligned} \tag{2.1}$$

where w is weight vector, b is the bias, y_i is label of data point x_i, l is the total number of data points, $c > 0$ is penalty parameter, and ξ_i is the slack variable.

The dual formulation of quadratic programming problem or QPP (2.1) is written by applying the Karush Kuhn Tucker (K.K.T.) conditions [35, 36] as

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ s.t. \quad & 0 \leq \alpha_i \leq c, \, \forall \, i = 1, 2, \ldots, l, \\ & \sum_{i=1}^{l}\alpha_i y_i = 0, \end{aligned} \tag{2.2}$$

where $\alpha_i \geq 0$ is the Lagrange multiplier [36], and $y_{i,j}$ is the class label.

The classifier is given as

$$f(x) = sgn(w^T x + b), \tag{2.3}$$

where sgn is the signum function, and the weight vector is calculated as $w = \sum_{i=1}^{l} \alpha_i y_i x_i$.

2.2 Twin support vector machine and class imbalance problem

The time complexity of solving QPP of SVM is $O(m^3)$ [12], where m is number of samples. To reduce the computational complexity of SVM, Jayadeva et al. [12] proposed an efficient twin support vector machine (TWSVM) for classification problems. In TWSVM, two hyperplanes are constructed instead of one as in SVM, and the optimization problem is to keep each of the hyperplanes closer to its own class and away from the other class. This leads to a time complexity of $2 * O(m/2)^3$ i.e., $O(m)^3/4$ in TWSVM. The TWSVM algorithm is one of the most prominent techniques for classification problems. It has been applied in various real world applications, due to its less computational complexity. Kumar and Gopal [18] proposed a more efficient least squares twin support vector machine (LSTSVM), where a pair of system of linear equations is solved. The computation time of LSTSVM is very less in comparison to SVM.

One of the important applications of SVM is the classification of class imbalance datasets. In most applications, there is an imbalance in the number of samples of the classes, leading to incorrect classification of data points in the minority class. Moreover, while dealing with imbalanced data, noisy data poses a major challenge in various applications. In many applications involving high imbalance in the data, such as disease [37], fault [38], and defective software modules detection [39], the priority is to correctly classify the minority class. For example, in disease detection there are very less samples of people with disease in comparison to healthy people.

In QPP of SVM, all the data points serve as constraints, but in TWSVM the data is distributed in such a way that one class gives the constraints to the other class and vice-versa. So, TWSVM solves two smaller size QPPs rather than one large QPP in SVM. The formulation of TWSVM is described in the following subsection.

2.2.1 Twin support vector machine (TWSVM)

Consider a binary classification problem where the data points belong to class +1 and -1, which are represented by matrices X_1 and X_2 respectively.

The optimization problems of TWSVM in the non-linear case [12] are written as:

$$\begin{aligned} \min_{w_1,\, b_1,\, \xi_1} \ & \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_1 \\ s.t. \quad & -(K(X_2, D^T)w_1 + e_2 b_1) + \xi_1 \geq e_2,\ \xi_1 \geq 0, \end{aligned} \tag{2.4}$$

$$\begin{aligned} \min_{w_2,\, b_2,\, \xi_2} \ & \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_2 \\ s.t. \quad & (K(X_1, D^T)w_2 + e_1 b_2) + \xi_2 \geq e_1,\ \xi_2 \geq 0, \end{aligned} \tag{2.5}$$

where $w_i, i = 1, 2$ is weight vector and b_i is bias of hyperplane of i^{th} class, ξ_i is slack variable, and $D = [X_1^T\ X_2^T]^T$. Here, $K(x^T, D^T) = (K(x, x_1), \ldots, K(x, x_l))$ is a row vector in $\mathbb{R}^n$, and e_i is vector of ones of appropriate dimension.

By using the K.K.T. necessary and sufficient conditions, the Wolfe duals of Eqs. (2.4) and (2.5) are obtained as

$$\begin{aligned} \max_{\alpha} \ & e_2^T \alpha - \frac{1}{2}\alpha^T N (M^T M)^{-1} N^T \alpha \\ s.t. \ & 0 \leq \alpha \leq c_1, \end{aligned} \tag{2.6}$$

$$\begin{aligned} \max_{\beta} \ & e_1^T \beta - \frac{1}{2}\beta^T M (N^T N)^{-1} M^T \beta \\ s.t. \ & 0 \leq \beta \leq c_2, \end{aligned} \tag{2.7}$$

where $M = [K(X_1,\ D^T)\ e_1]$ and $N = [K(X_2,\ D^T)\ e_2]$, α and β are vectors containing the Lagrange multipliers. The classifying hyperplanes $K(x^T, D^T)w_1 + b_1 = 0$ and $K(x^T, D^T)w_2 + b_2 = 0$ are constructed by using the values of w_i, and b_i, $i = 1, 2$ from the following equations,

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(M^T M + \delta I)^{-1} N^T \alpha, \tag{2.8}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (N^T N + \delta I)^{-1} M^T \beta, \tag{2.9}$$

where $\delta > 0$ is a small positive value to avoid ill-conditioning of the matrices $M^T M$ and $N^T N$ in calculating the inverse, and I is identity matrix of appropriate size.

A new data point $x \in \mathbb{R}^n$ is classified using the following decision function,

$$class(i) = min(|K(x^T, D^T)w_i + b_i|) \quad \text{for} \quad i = 1, 2, \tag{2.10}$$

where $|.|$ is the perpendicular distance of point x from the hyperplane. Similarly, for the linear case, the decision function can be given as

$$class(i) = min(|w_i^T x + b_i|) \quad \text{for} \quad i = 1, 2. \tag{2.11}$$

2.2.2 Least squares twin support vector machine (LSTSVM)

The QPPs of LSTSVM [18] for non-linear case are described as

$$\begin{aligned} \min_{w_1, b_1, \eta_1} \quad & \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + \frac{c_1}{2}\eta_1^T \eta_1 \\ s.t. \quad & -(K(X_2, D^T)w_1 + e_2 b_1) + \eta_1 = e_2, \end{aligned} \tag{2.12}$$

$$\begin{aligned} \min_{w_2,b_2,\eta_2} \quad & \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + \frac{c_2}{2}\eta_2^T \eta_2 \\ s.t. \quad & K(X_1, D^T)w_2 + e_1 b_2 + \eta_2 = e_1, \end{aligned} \tag{2.13}$$

where $c_i, i = 1, 2$ are positive parameters, $\eta_i, i = 1, 2$ denote the slack variables, $K(., D^T)$ is the kernel matrix where $D = [X_1^T \; X_2^T]^T$, and e_1, e_2 represent vectors of ones of suitable dimensions.

By using the constraints in their respective objective functions, we get

$$\min_{w_1,b_1} \quad \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + \frac{c_1}{2}\|K(X_2, D^T)w_1 + e_2 b_1 + e_2\|^2, \tag{2.14}$$

$$\min_{w_2,b_2} \quad \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + \frac{c_2}{2}\| - (K(X_1, D^T)w_2 + e_1 b_2) + e_1\|^2. \tag{2.15}$$

Taking the gradient of QPP (2.14) w.r.t. w_1 and b_1, we get

$$K\big(X_1, D^T\big)^T\big(K(X_1, D^T)w_1 + e_1 b_1\big) + c_1 K\big(X_2, D^T\big)^T\big(K(X_2, D^T)w_1 + e_2 b_1 + e_2\big) = 0, \tag{2.16}$$

$$e_1^T\big(K(X_1, D^T)w_1 + e_1 b_1\big) + c_1 e_2^T\big(K(X_2, D^T)w_1 + e_2 b_1 + e_2\big) = 0. \tag{2.17}$$

Combining Eqs. (2.16) and (2.17) and solving, we get

$$[w_1 \;\; b_1]^T = -\left(G^T G + \frac{1}{c_1} H^T H\right)^{-1} G^T e_2, \tag{2.18}$$

where $H = [K(X_1, D^T) \;\; e_1]$, and $G = [K(X_2, D^T) \;\; e_2]$. Similarly, using Eq. (2.15), we get

$$[w_2 \;\; b_2]^T = \left(H^T H + \frac{1}{c_2} G^T G\right)^{-1} H^T e_1. \tag{2.19}$$

For reducing the computation time of finding the inverse, Sherman–Morrison–Woodbury (SMW) formula [40] is used for Eqs. (2.18) and (2.19). A testing data point x is assigned to a class using Eq. (2.10).

2.2.3 Fuzzy membership functions

In [41], Lin and Wang proposed a fuzzy support vector machine (FSVM) based on distance from the class centroid for each class. This reduces the effect of outliers in the classification because the outliers get relatively less weight for the classification as compared to the other points. In class imbalance learning, due to a huge difference in the number of samples of the binary classes, SVM classifier gives more priority to the samples of the majority class, and misclassifies the samples which are in minority. To give more weight to the minority class, different weights are assigned to the data points of both the classes. Since FSVM is not suitable for class imbalance learning, Batuwita and Palade [42] proposed FSVM-CIL with different settings of parameters and fuzzy membership functions. An improved one-class SVM for class imbalance is proposed in [43] using a conformal kernel transformation. A boosting algorithm for support vector machine [44] is proposed for countering the excessive bias in classifying imbalance data. FSVM for class imbalance in medical datasets is proposed [45] for incorporating the local information using a local within-class preserving scatter matrix. A scaling kernel function is proposed [46] for SVM in class imbalance learning. An oversampling technique is combined with the undersampling technique in a hybrid sampling approach for SVM [47].

A weighted least squares projection twin support vector machines is proposed in [48] to include the local information about the data. A fuzzy least squares twin support vector machine is proposed [49] to deal with class imbalance datasets. For class imbalance data with missing values, Razzaghi et al. [50] proposed a multilevel framework of the cost-sensitive SVM for healthcare data. Moreover, a fuzzy total margin based support vector machine (FTM-SVM) is proposed with different settings in [51] for imbalance problems. A weighted K-means support vector machine for cancer prediction is proposed in [52] to circumvent the problem of imbalance in the data. Further, a weighted least squares twin support vector machine (WLSTSVM) is proposed for binary classification in [53], while a weighted multi-class least squares twin support vector machine (WMLSTSVM) is proposed in [54]. In WMLSTSVM,

the fuzzy membership function gives membership on the basis of number of samples in the two classes, and thus is not capable of dealing with outliers. Recently, an entropy based fuzzy support vector machine (EFSVM) is proposed in [55]. In EFSVM, the data points of the majority class are given fuzzy membership based on their information entropy on the basis of proximity to the binary classes.

Here, we discuss some of the fuzzy membership functions used for class imbalance. The fuzzy membership functions for centroid based membership [42] are as follows:
I. Centroid (linear): The fuzzy membership is assigned based on the distance of the data points from the centroid of its class. Here, the decaying function is linear in nature. The fuzzy membership function is given as

$$mem = 1 - \left(\frac{d_{cen}}{max(d_{cen}) + \delta} \right), \tag{2.20}$$

where d_{cen} is the Euclidean distance of each data point from the centroid of its class, and δ is a small positive integer to remove the possibility of division by 0.
II. Centroid (exponential): The decaying function is exponential in nature and the fuzzy membership is assigned based on the distance of the data points from the centroid of its class. The fuzzy membership function is written as

$$mem = \left(\frac{2}{1 + exp(\beta d_{cen})} \right), \tag{2.21}$$

where d_{cen} is the Euclidean distance of each data point from the centroid of its class, and β decides the scale of the exponential function.

2.3 Universum learning and its applications

Weston et al. [10] proposed a universum support vector machine (USVM) to give prior information to the classifier about the distribution of data. The universum data points do not belong to any of the classes, and lie within a tube between the two classes. This approach is also called as 'learning through contradiction'. In USVM, along with the hinge loss it involves an ϵ-insensitive loss function. This universum based approach

has been applied to various real world applications. Long and Tang [22] performed the classification of investor sentiments using USVM. Gao et al. [56] used universum SVM for prediction of translation initiation in proteins. They used two approaches for selecting the universum: one is based on uniform distribution of noise and other using random averaging of the data points. Hao and Zhang [57] proposed an ensemble universum support vector machine for the detection of Alzheimer's disease from brain imaging data by using the patients with mild cognitive impairment (MCI) as the universum. In the following subsection, we describe the formulation of USVM.

2.3.1 Universum support vector machine (USVM)

The optimization problem of USVM is given as follows:

$$\begin{aligned} \min_{w,b,\xi,\eta} \quad & \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{l}\xi_i + c_u\sum_{j=1}^{2r}\eta_j \\ s.t. \quad & y_i(w^T\phi(x_i)+b) \geq 1-\xi_i, \\ & y_j(w^T\phi(x_j)+b) \geq -\epsilon-\eta_j, \\ & \xi_i \geq 0,\ \eta_j \geq 0,\ \forall i = 1,2,\ldots,l,\ \forall j = 1,2,\ldots,2r, \end{aligned} \tag{2.22}$$

where l is the total number of data points, $c > 0, c_u > 0$ are penalty parameters, ξ_i and η_j are slack variables, ϵ is the parameter for the insensitive tube, $\phi : R^n \rightarrow R^p$ is the function mapping from n to p dimension where $p > n$, and r is the number of universum samples.

The dual of Eq. (2.22) is written by applying the K.K.T. conditions as,

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l+2r}\mu_i\alpha_i - \frac{1}{2}\sum_{i=1}^{l+2r}\sum_{j=1}^{l+2r}\alpha_i\alpha_j y_i y_j \phi(x_i)^T\phi(x_j) \\ s.t. \quad & 0 \leq \alpha_i \leq c,\ \mu_i = 1,\ \forall\, i = 1,2,\ldots,l, \\ & 0 \leq \alpha_i \leq c_u,\ \mu_i = -\epsilon,\ \forall\, i = l+1, l+2,\ldots,l+2r, \\ & \sum_{i=1}^{l+2r}\alpha_i y_i = 0, \end{aligned} \tag{2.23}$$

where $\alpha_i \geq 0$ is the Lagrange multiplier.

The classifier shown in Fig. (1.2) is given by Eq. (2.3), and the weight vector is obtained as $w = \sum_{i=1}^{l+2r} \alpha_i y_i x_i$.

2.3.2 Classification of EEG signals using SVM

Electroencephalogram (EEG) signal classification is a major challenge in the field of machine learning and signal processing. EEG is a widely used non-invasive technique for the detection of various types of brain disorders, such as epileptic seizures and sleep disorders. In epilepsy, the extent of disease ranges from partial to generalized seizures which are reflected in their respective EEG. The different types of EEG signals are shown in Fig. 2.1. For the better feature extraction and classification of EEG signals, several signal processing techniques have been used by researchers. Among the various feature extraction techniques, wavelet transform is one of the frequently used methods. In wavelet transform, the frequency domain features are extracted from the signal with good localization in time. This is in contrast to the Fourier transform, where the signal analysis is done mainly in the frequency domain. In wavelet analysis, the approximation and decomposition coefficients are used to form the feature vector as shown in Fig. 2.2.

The different families of wavelet are used for specific type of signals to get better characteristics of that signal. Adeli et al. [58] proposed a computer aided diagnosis (CAD) method for epilepsy using discrete wavelet transform (DWT). They used Daubechies wavelet with db-4 as the mother wavelet for the feature extraction. An orthogonal decimated discrete wavelet transform (ODWT) [59] is used for detecting maturational changes associated with childhood absence epilepsy. The classification of EEG signals is performed [60] using wavelet packet analysis and genetic algorithm. Daubechies wavelet-2 is used for the classification of five different EEG signals [61]. Subasi and Gursoy [23] used principal component analysis (PCA), linear discriminant analysis (LDA) and independent component analysis (ICA) for the feature extraction, and SVM for classification.

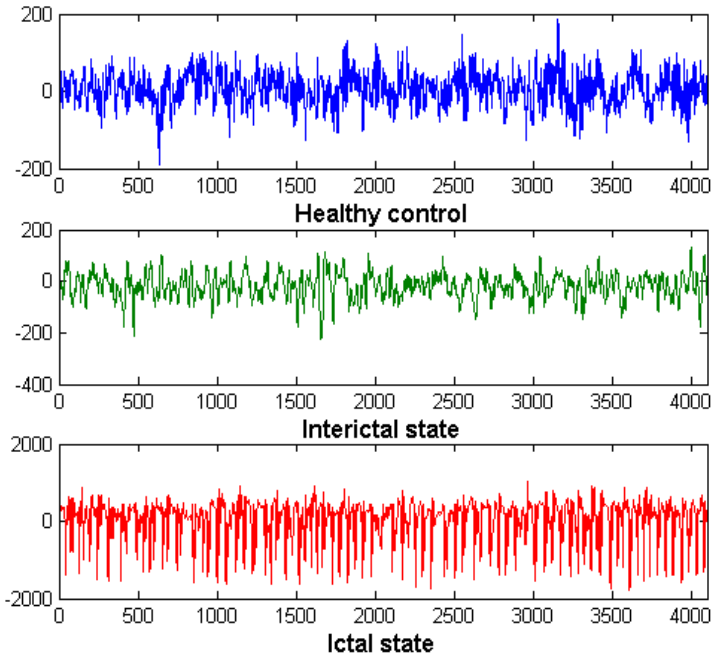


Figure 2.1: Healthy, interictal, and ictal EEG signals.

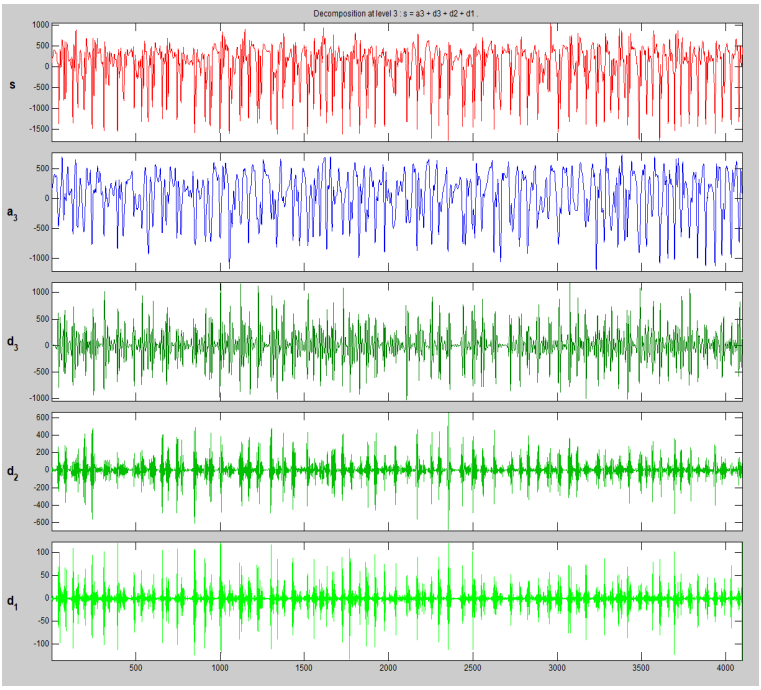


Figure 2.2: Discrete wavelet decomposition of EEG signal at 3rd level of decomposition using Daubechies-4 wavelet.

The proper selection of classification techniques is very crucial for the automated diagnosis of patients having neurological diseases. Among the various classification algorithms, support vector machines (SVMs) have emerged as a powerful classification technique. SVM solves a convex optimization problem which leads to a globally optimal solution. This is in contrast to artificial neural network (ANN) that suffers from the problem of local minima. SVM also has a lower VC (Vapnik-Chervonenkis) dimension that enables it to classify high dimensional data with less optimizing parameters.

Many researchers have used SVM in the classification of EEG signals [62] and for the diagnosis of neurological diseases like epilepsy [63–65]. Guo et al. (2010) [66] performed the classification of mental tasks from the analysis of EEG signals using SVM. Least squares support vector machine (LSSVM) [17] is also used [67, 68] for the detection of epilepsy. LSSVM is used for classification of EEG signal with a clustering based approach [69]. For multiclass classification of EEG signals, Guler and Ubeyli (2007) [70] proposed a SVM based model and showed that SVM gives better classification accuracy for EEG signals as compared to probabilistic neural network (PNN) and multilayer perceptron neural network (MLPNN).

The models proposed in this thesis for neurological disorders also involve efficient universum SVM based models. Therefore, in the following subsections, we discuss the formulations of some existing universum SVM based algorithms, which are improvements over USVM in terms of computation time. An efficient universum based technique known as universum twin support vector machine (UTSVM) is discussed in the following subsection.

2.3.3 Universum twin support vector machine (UTSVM)

Universum based algorithms give better generalization performance [10, 19] due to inclusion of universum data in the optimization problem. However, the model becomes computationally expensive [20,71], due to a single large QPP with additional universum data. To remove this drawback, a universum twin support vector machine (UTSVM) is proposed [27]. UTSVM solves two smaller size QPPs instead of one large

QPP. This makes UTSVM computationally faster than USVM, as TWSVM is faster than SVM. Here, the additional constraints in each QPP are for the universum data points.

The optimization problems of UTSVM in the non-linear case are written as follows:

$$\begin{aligned}\min_{w_1,\, b_1,\, \xi_1,\, \eta_1} \ & \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_1 + c_u e_u^T \eta_1 \\ s.t. \quad & -(K(X_2, D^T)w_1 + e_2 b_1) + \xi_1 \geq e_2, \\ & (K(U, D^T)w_1 + e_u b_1) + \eta_1 \geq (-1+\epsilon)e_u, \\ & \xi_1 \geq 0, \quad \eta_1 \geq 0, \end{aligned} \tag{2.24}$$

$$\begin{aligned}\min_{w_2,\, b_2,\, \xi_2,\, \eta_2} \ & \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_2 + c_u e_u^T \eta_2 \\ s.t. \quad & (K(X_1, D^T)w_2 + e_1 b_2) + \xi_2 \geq e_1, \\ & -(K(U, D^T)w_2 + e_u b_2) + \eta_2 \geq (-1+\epsilon)e_u, \\ & \xi_2 \geq 0, \quad \eta_2 \geq 0, \end{aligned} \tag{2.25}$$

where $c_i (i = 1, 2)$ and c_u are positive real penalty parameters; ξ_i, $\eta_i (i = 1, 2)$ are slack variables, and $e_i (i = 1, 2)$, e_u are vectors of ones of suitable dimensions.

By applying the K.K.T. necessary and sufficient conditions, the Wolfe duals of Eqs. (2.24) and (2.25) are obtained as

$$\begin{aligned}\max_{\alpha_1,\, \mu_1} \ & e_2^T \alpha_1 - \frac{1}{2}(\alpha_1^T N - \mu_1^T O)(M^T M)^{-1}(N^T \alpha_1 - O^T \mu_1) + (\epsilon - 1)e_u^T \mu_1 \\ s.t. \ & 0 \leq \alpha_1 \leq c_1, \quad 0 \leq \mu_1 \leq c_u \end{aligned} \tag{2.26}$$

$$\begin{aligned}\max_{\alpha_2,\, \mu_2} \ & e_1^T \alpha_2 - \frac{1}{2}(\alpha_2^T M - \mu_2^T O)(N^T N)^{-1}(M^T \alpha_2 - O^T \mu_2) + (\epsilon - 1)e_u^T \mu_2 \\ s.t. \ & 0 \leq \alpha_2 \leq c_2, \quad 0 \leq \mu_2 \leq c_u, \end{aligned} \tag{2.27}$$

where $M = [K(X_1, D^T)\ e_1]$, $N = [K(X_2, D^T)\ e_2]$ and $O = [K(U, D^T)\ e_u]$; α_1, α_2,

μ_1, μ_2 are vectors of Lagrange multipliers.

The classifying hyperplanes $K(x^T, D^T)w_1 + b_1 = 0$ and $K(x^T, D^T)w_2 + b_2 = 0$ are constructed from the parameter values of $w_i (i = 1, 2)$ and $b_i (i = 1, 2)$ using the following Eqs. (2.28) and (2.29),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(M^T M)^{-1}(N^T \alpha_1 - O^T \mu_1), \tag{2.28}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (N^T N)^{-1}(M^T \alpha_2 - O^T \mu_2). \tag{2.29}$$

Note that the matrices $M^T M$ and $N^T N$ are always positive semi-definite, it is possible that they may not be well conditioned in some situations. So, a regularization term δI, $\delta > 0$ is introduced with the matrices $M^T M$ and $N^T N$ as $(M^T M + \delta I)$ and $(N^T N + \delta I)$. Here, I is an identity matrix of appropriate dimension. Each new data point is classified using Eq. (2.10).

In the following section, we discuss another efficient formulation for universum based SVM algorithms, known as least squares twin support vector machine with universum data (ULSTSVM).

2.3.4 Least squares twin support vector machine with universum data (ULSTSVM)

A least squares twin support vector machine with universum data (ULSTSVM) [71] is proposed to reduce the computation time of UTSVM. The optimization problem of ULSTSVM [27] comprises the following QPPs,

$$\begin{aligned} \min_{w_1, b_1, \xi_1, \eta_1} \quad & \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + \frac{c_1}{2}\xi_1^T \xi_1 + \frac{c_3}{2}(\|w_1\|^2 + b_1^2) + \frac{c_5}{2}\eta_1^T \eta_1 \\ s.t. \quad & -(K(X_2, D^T)w_1 + e_2 b_1) + \xi_1 = e_2, \\ & K(U, D^T)w_1 + e_u b_1 + \eta_1 = (-1 + \epsilon)e_u, \end{aligned} \tag{2.30}$$

$$\begin{aligned}\min_{w_2,b_2,\xi_2,\eta_2} \quad & \frac{1}{2}\|K(X_2,D^T)w_2+e_2b_2\|^2+\frac{c_2}{2}\xi_2^T\xi_2+\frac{c_4}{2}(\|w_2\|^2+b_2^2)+\frac{c_6}{2}\eta_2^T\eta_2\\ s.t. \quad & K(X_1,D^T)w_2+e_1b_2+\xi_2=e_1,\\ & -(K(U,D^T)w_2+e_ub_2)+\eta_2=(-1+\epsilon)e_u, \end{aligned} \tag{2.31}$$

where $\xi_i,\eta_i,i=1,2$ represent slack variables, $c_i,i=1,2$ are positive parameters, and $K(.,D^T)$ is the kernel matrix and $D=[X_1^T\ X_2^T]^T$.

Substituting the constraints in the objective functions, we get

$$\begin{aligned}\min_{w_1,b_1} \quad & \frac{1}{2}\|K(X_1,D^T)w_1+e_1b_1\|^2+\frac{c_1}{2}\|K(X_2,D^T)w_1+e_2b_1+e_2\|^2+\frac{c_3}{2}(\|w_1\|^2+b_1^2)\\ & +\frac{c_5}{2}\|-(K(U,D^T)w_1+e_ub_1)+(-1+\epsilon)e_u\|^2, \end{aligned} \tag{2.32}$$

$$\begin{aligned}\min_{w_2,b_2} \quad & \frac{1}{2}\|K(X_2,D^T)w_2+e_2b_2\|^2+\frac{c_2}{2}\|-(K(X_1,D^T)w_2+e_1b_2)+e_1\|^2+\frac{c_4}{2}(\|w_2\|^2+b_2^2)\\ & +\frac{c_6}{2}\|(K(U,D^T)w_2+e_ub_2)+(-1+\epsilon)e_u\|^2. \end{aligned} \tag{2.33}$$

Taking the gradient of Eq. (2.32) w.r.t. w_1 and b_1 and equating to 0, we get

$$\begin{aligned}K\big(X_1,D^T\big)^T\big(K(X_1,D^T)w_1+e_1b_1\big)+c_1K\big(X_2,D^T\big)^T\big(K(X_2,D^T)w_1+e_2b_1+e_2\big)+c_3w_1\\ +c_5K(U,D^T)^T\big(K(U,D^T)w_1+e_ub_1-(-1+\epsilon)e_u\big)=0,\end{aligned} \tag{2.34}$$

$$\begin{aligned}e_1^T\big(K(X_1,D^T)w_1+e_1b_1\big)+c_1e_2^T\big(K(X_2,D^T)w_1+e_2b_1+e_2\big)+c_3b_1\\ +c_5e_u^T\big(K(U,D^T)w_2+e_ub_1-(-1+\epsilon)e_u\big)=0.\end{aligned} \tag{2.35}$$

Combining Eqs. (2.34) and (2.35) and solving, we get

$$[w_1\ \ b_1]^T=-\big(H^TH+c_1G^TG+c_3I+c_5O^TO\big)^{-1}\big(c_1G^Te_2+c_5(1-\epsilon)O^Te_u\big), \tag{2.36}$$

and using Eq. (2.33), we get

$$[w_2 \;\; b_2]^T = \big(G^T G + c_2 H^T H + c_4 I + c_6 O^T O\big)^{-1} \big(c_2 H^T e_1 + c_6(1-\epsilon) O^T e_u\big). \quad (2.37)$$

where $H = [K(X_1, D^T) \;\; e_1]$, $G = [K(X_2, D^T) \;\; e_2]$, and $O = [K(U, D^T) \;\; e_u]$.

For a new data point x, the class assignment is performed using Eq. (2.10).

2.4 Diagnosis of Alzheimer's disease using machine learning techniques

Alzheimer's disease (AD) is one of the most common cause of dementia in today's world. According to World Alzheimer Report (2018) [11], around 50 million people were affected by this disease in 2018, which is expected to triple by 2050. Usually, the symptoms of AD are visible after 60 years of age [72]. However, some forms of AD develop very early (30-50 years) for individuals having gene mutation [73]. Alzheimer's disease gives rise to structural and functional changes in the brain. In AD patients, the time between healthy state to Alzheimer's spans over many years [74]. First, patients develop mild cognitive impairment (MCI), and gradually progress to AD. However, all MCI patients do not convert to AD [75]. So, the main focus of current research is to predict the conversion of MCI to AD. These changes can be measured using medical imaging [76] and other techniques like blood plasma spectroscopy [77, 78].

Many open source databases for Alzheimer's disease have accelerated research in this field [79, 80]. The most widely used databases are ADNI [81] (`adni.loni.usc.edu`), AIBL (`aibl.csiro.au`), OASIS (`www.oasis-brains.org`). A new publicly available database for clinical Alzheimer's data is J-ADNI database [82,83] containing data from longitudinal studies in Japan. Further, processing of MRI images requires a lot of effort. To facilitate analysis of MRI images open source softwares like Statistical Parametric Mapping (SPM) have been developed by Wellcome Centre for Human Neuroimaging for public use. SPM is used for voxel based morphometry (VBM) [84] of MRI data. Another very popular open source software i.e., Freesurfer [85] is developed

for volume based morphometry and is used by many researchers [86, 87].

Machine learning techniques are found to be very useful for the diagnosis of AD [88–90] in the last decade. The most widely used classification techniques are support vector machine (SVM), artificial neural network (ANN), and deep learning. The primary difference between SVM and ANN is the nature of the optimization problem. SVM gives a globally optimal solution [91], while ANN gives locally optimal solution. In both SVM and ANN, feature extraction is an important step. Shi et al. [92] suggested that combination of neural networks and intelligent agents can be useful for medical image analysis. However, deep learning incorporates the feature extraction step in the learning model itself [93,94]. For large datasets, deep learning is found to be useful especially for image data [93]. Some researchers also used ensemble methods to improve the classification accuracy for AD [95–97].

For classification of AD data, the accuracy is dependent on the type of problem. For example, the accuracy is highest for Control normal (CN) vs AD, lesser for CN vs MCI, and least for MCI vs AD [98]. Moreover, the classification of MCI converters (MCIc) vs non-converters (MCInc), and amnestic MCI (aMCI) vs non-amnestic MCI (naMCI) is also a challenging task [99, 100]. Moreover, the data generated from MRI scanners is 3-D in nature and thus amounts to large sized datasets. So, efficient feature extraction and classification techniques are needed to analyze this data [101, 102].

2.4.1 Search strategy

We searched prominent papers in the field from Google Scholar (`https://scholar.google.co.in`) and Sciencedirect (`https://www.sciencedirect.com`). We excluded the studies which did not use accuracy measures for classification performance. This resulted in a total of 165 papers. Out of 165, 60 papers used SVM, 45 used a combination of ANN, multi-task learning, transfer learning, multi-kernel learning and certain feature selection techniques. We also reviewed 60 papers based on deep learning and ensemble methods for AD.

As per our survey [2], it is found that SVM based models have been widely used for Alzheimer's disease showing its robustness. This is because techniques like ANN

suffers from the drawbacks of local minima, which is not the case with SVM. Therefore, in the following sections, we present a review on the works using SVM for Alzheimer's disease. The review is summarized in Table 2.1 with details of these works.

2.4.2 Image modality

Image modality is a prominent factor for classification of MRI images. In case of structural MRI (sMRI) images, most of the researchers used T1-weighted images while only few researchers used T2 images [103–105]. This is because the delineation of ventricular surface of brain due to atrophy is clearly visible in T1-weighted images [24].

Fan et al. [106] suggested that positron emission tomography (PET) scans provide complementary information to sMRI scans, thus improving the classification accuracy of CN vs MCI using SVM. Dukart et al. [107] supported this fact that fluorodeoxyglucose-PET (FDG-PET) features are more discriminative as compared to sMRI. Further, better accuracy is found for CN vs AD [108] with PET images (100 %) as compared to single photon emission computed tomography (SPECT) images (97.5 %). Similar finding is observed for CN vs AD [109] with better accuracy for PET images (96.67 %) as compared to SPECT images (94.5 %). Kamathe et al. [105] used combination of T1, T2 and proton density (PD) scans for classification of CN vs AD. Hojjati et al. [110] used resting state functional MRI (rs-fMRI) to find the connectivity changes in brain for classification of MCIc vs MCInc, while Sheng et al. [111] used connectivity information from fMRI data. Fig. 2.3 shows the usage of different modalities of data for SVM in our survey.

Diffusion tensor imaging (DTI) is also explored by various researchers for Alzheimer's disease [100, 112, 113]. Haller et al. [114] found that SVM based analysis of white matter DTI parameters is helpful in classification of different types of MCI patients.

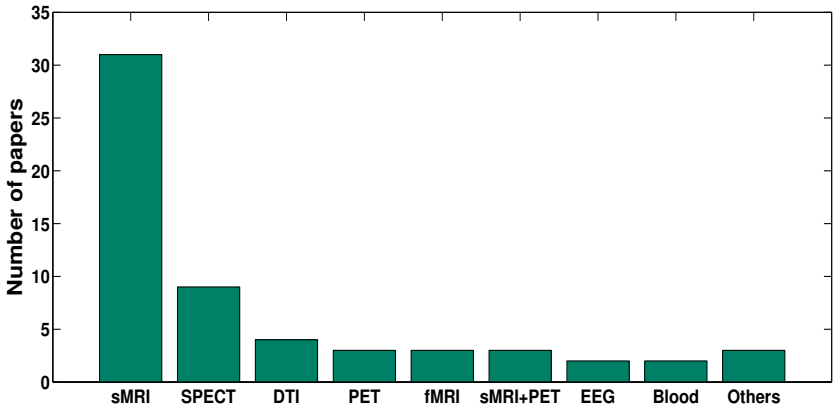


Figure 2.3: Plot showing different image modalities and other data used with SVM for AD.

2.4.3 Feature selection and extraction with SVM

Feature selection plays an important part in the classification of data. Different features are combined to form the feature vector in many works [110,115,116]. Vemuri et al. [117] found that including demographic and genetic information with sMRI scans improved the classification accuracy of CN vs AD. A refined parcellation method is proposed [118] for detecting subtle changes in gray matter (GM). Magnin et al. [7] presented a feature selection method based on histogram of regions of interests (ROIs) for CN vs AD. Gerardin et al. [119] used shape features of hippocampus to discriminate CN, MCI and AD, and found that shape deformation features are better than volumetric features. Normalized mean square error (NMSE) features are used [120] to discriminate CN with early AD. A clustering based approach is proposed [97] to group adjacent voxels for classification of CN, MCI and AD. Fisher discriminate ratio (FDR) is used [121] to extract useful voxels as features (VAF) from SPECT images.

Gaussian mixture model (GMM) is used in [109] for CN vs AD. It is stated that the proposed GMM based feature extraction makes the data linearly separable. Ortiz et al. [122] used PET and sMRI data to find the most discriminative features using sparse inverse covariance estimation (SICE) method with SVM. Non-negative matrix

factorization (NMF) based features with SVM are found to give better performance than PCA with SVM to classify CN vs AD [123]. Moreover, Abdulkadir et al. [124] illustrated the affects of hardware heterogeneity on classification accuracy of SVM. They also found high confidence level of classification performance for large samples. Cuingnet et al. [99] stated that DARTEL based features are better than SPM features for CN, MCIc, MCInc, and AD. Moreover, it is concluded that feature selection techniques for sMRI images may lead to less classification accuracy due to addition of hyperparameters. Further, Schmitter et al. [125] found volume based features to be more useful than voxel based morphometry (VBM). Morphological features of brain regions are used by Plocharsky et al. [126] to classify CN vs AD, while Long et al. used shape differences in the subjects' brains for classification of CN, AD, sMCI (stable), and pMCI (progressive). Fuzzy based classes for hippocampus volume are used by Tangaro et al. [87] for classification of CN vs AD, and MCIc vs MCInc.

Wavelet based features are used in various works. Chaplot et al. [103] used discrete wavelet transform (DWT) features, while Zhang et al. [116] found that 3-D DWT and SVM are useful for classification of CN, MCI and AD subjects. Segovia et al. [127] discovered that partial least squares (PLS) components have a higher FDR score as compared to principal component analysis (PCA) for CN vs AD using SPECT images. Ortiz et al. used self organizing maps (SOMs) [115] for unsupervised segmentation of sMRI images in classification of CN vs AD. However, Chaplot et al. [103] found that SVM performs better than SOM for classification of AD patients using T2-weighted images.

Other techniques like SVM-RFE [1,128] are used as an optimized feature selection technique in [129] to select prominent brain features for CN vs AD. Independent component analysis (ICA) is used in many works [130, 131] for classification of CN vs AD using SVM. EEG data is also used [132] for classification of CN vs AD using SVM. Mazaheri et al. [133] used EEG recordings of word comprehension by subjects to classify MCIc from MCInc and CN. Some researchers also focused on blood based biomarkers for AD [77,78]. Gostolya et al. [134] used speech patterns of subjects and classified using linear SVM.

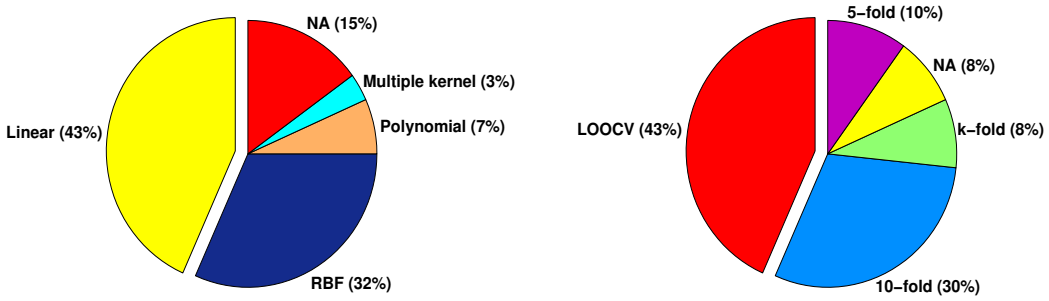
2.4.4 Kernel functions for AD

Different kernels have been used with SVM for classification of AD. Various researchers have used linear kernel with SVM to classify Alzheimer's data as shown in Fig. 2.4(a). This is due to the fact that in linear kernel, there is no kernel parameter to tune. Some researchers also utilized multiple kernels for SVM [86]. Moreover, in most papers, the sample size is also very small as shown in Table 2.1, which may lead to overfitting of the data with radial basis function (RBF) kernel [128]. This leads to the use of linear kernel due to its simplicity. The usage of different kernels as per our survey is shown in Fig. 2.4(a).

Kloppel et al. [135] used linear SVM to classify pathologically confirmed cases of AD with CN, and suggested that SVM can help in the diagnosis of AD. It has been stated in [121,131] that linear kernel provides better classification performance for high dimensional data as compared to polynomial or RBF kernel. However, polynomial kernel is also used by researchers. Lahmiri et al. [104] used polynomial kernel for multiclass classification of CN, MCI, and AD. Zhang et al. [136] found that polynomial kernel is useful in classification of CN vs AD using PCA features. In 2018, Lahmiri et al. [137] used volumetric features with cognitive test scores for classification of CN vs AD with polynomial kernel.

Some researchers also used an ensemble of kernels. Multiple kernel SVM is used by Alam et al. [86] for classification of CN, MCI, and AD. Kamathe et al. [105] used linear, polynomial and RBF kernel for classification of CN vs AD. Peng et al. [138] used MRI and genetic data for features, and used multiple kernel learning with SVM to classify the subjects. The selection of optimal hyperparameters is a major step in the classification of SVM. Among the various methods, leave one out cross validation (LOOCV) has been widely used for classification of AD using SVM. The details of the cross-validation methods are shown in Fig. 2.4(b).

In the following subsection, we present a comprehensive survey on the usage of different variants of SVM for AD.



(a) (b)

Figure 2.4: (a) Plot showing usage of different types of kernels and (b) cross validation methods used with SVM for AD. NA means information about kernel is not available, and $k = 2, 3, 4, 9$ and 20.

2.4.5 Variants of SVM used for AD

Many variants of SVM are developed for different types of classification problems. In the formulation of SVM, there is no spatial information of the brain image in the optimization problem [139, 140]. To provide spatial information, contiguous SVM (CSVM) is used to classify SPECT images of AD and control subjects [139, 141]. CSVM uses the information about voxel connectivity to give a more robust classifier. For reducing the computation cost, Zhang et al. [142] used twin support vector machine (TWSVM) for classification of CN vs AD, while structural least squares twin support vector machine (S-LSTSVM) is used in [143]. For optimized feature selection, Beheshti et al. [98] used genetic algorithm (GA) with linear SVM for classification of CN vs AD and pMCI vs sMCI.

For early diagnosis of AD, Zhu et al. [102] used a temporally structured SVM (TS-SVM) for classification of longitudinal MR images of MCI converters and non-converters. Lu et al. [144] proposed a random forest robust SVM (RF-RSVM) for classification of CN vs MCI using FDG-PET images. TWSVM is used for classification of CN vs AD [145] using dual-tree complex wavelet transform (DTCWT), LDA and PCA features. Sun et al. [146] introduced spatial anatomical regularization with SVM for classification of CN, AD, sMCI, and pMCI. To optimize the SVM parameters, Zeng

et al. [147] proposed a switching delayed particle swarm optimization SVM (SDPSO-SVM). In order to reduce the complexity of SVM, Bi et al. [148] used random support vector machine clusters for classification of CN vs AD using rs-fMRI. In the random SVM approach, samples and features randomly are chosen randomly from the dataset and trained accordingly. It helps to reduce the size of training data leading to less computational complexity. Some researcher also applied ensemble based SVM for better prediction accuracy in AD classification [108, 149, 150].

The usage of different cross validation strategies is shown in Fig. 2.4(b). LOOCV comes out to be the frequently used method. This may be attributed to small sample size in the works shown in Table 2.1.

2.4.6 Observations

In the classification of dementia related data, there are various categories or targets. One classification target is MCI vs AD, which is one of the most important targets for early diagnosis of AD. It can be observed in Fig. 2.5(b) that most of the work has been done in classification of CN vs AD and CN vs MCI. Moreover, classifications like MCI vs AD are very less. This needs to be addressed in future research for early detection of AD. Other categories like MCIc vs MCInc are also addressed in very few papers. Therefore, researchers can focus on these particular problems for early detection of dementia caused by Alzheimer's disease.

The usage of different types of SVM in our survey is shown in Fig. 2.5(a). One can notice that among the different variants of SVM, 83% of the papers used standard SVM. This shows the popularity and robustness of SVM in the classification of MRI data [154]. In 3% of the papers, TWSVM is used [142, 145], whereas LSTSVM [143] is used in only 1 paper. The CSVM algorithm is used in [139, 141].

Some papers used ensemble of SVMs to classify Alzheimer's data [108, 149]. In AD diagnosis, an important focus point for research is the development of individual specific diagnosis models. For this, multimodal clinical data can be utilized as per the population. Moreover, novel learning techniques need to be developed for small datasets, since in real world scenarios the sample size from some population may not

Table 2.1: Comparison of research on classification of Alzheimer's data using SVM.

Sr. No.	Year	Authors	Target	Modality	Feature extraction	Machine learning	Dataset	Validation	Performance		
									Acc (%)	Sens (%)	Spec (%)
1	2005	Stoeckel et al. [139]	CN vs AD	SPECT	VBM+Spatial normalization	CSVM (Linear)	130 (31 CN, 99 AD)	LOOCV	72.5	93	-
2	2006	Chaplot et al. [103]	CN vs AD	sMRI (T2)	DWT	SVM (RBF)	52 (6 CN, 46 AD)	-	98	-	-
3	2007	Fung et al. [141]	CN vs AD	SPECT	Subsampling	CSVM (Linear)	130 (31 CN, 99 AD)	LOOCV	-	84.4	90.9
4	2008	Alvarez et al. [108]	CN vs AD	SPECT	Component based feature extraction	Ensemble SVM	79 (41 CN, 38 AD)	LOOCV	97.5	-	-
				PET			60 (18 CN, 42 AD)		100		
5	2008	Ramirez et al. [121]	CN vs AD	SPECT	FDR	SVM (RBF)	52 (23 CN, 29 AD)	-	90.38	-	-
6	2008	Gonzalez et al. [150]	CN vs AD	SPECT	Component based feature extraction	SVM (Linear)-Classification tree	79 (41 CN, 38 AD)	LOOCV	90	-	-
7	2008	Vemuri et al. [117]	CN vs AD	sMRI (T1)	VBM	SVM (Linear)	380 (190 CN, 190 AD)	4-fold	89.30	-	-
8	2008	Fan et al. [106]	CN vs MCI	sMRI+PET	RAVEN maps+ODC	SVM (Linear)	30 (15 CN, 15 MCI)	LOOCV	100	-	-
9	2008	Kloppel et al. [135]	CN vs AD	sMRI (T1)	VBM	SVM (Linear)	68 (34 CN, 34 AD)	LOOCV	95.6	97.1	94.1
			CN vs mAD				90 (57 CN, 33 AD)		85.6	75.8	91.2
10	2008	Mesrob et al. [118]	CN vs AD	sMRI (T1)	SVM-RFE	SVM (RBF)	61 (28 CN, 33 AD)	LOOCV	90.2	-	-
11	2009	Magnin et al. [7]	CN vs AD	sMRI (T1)	VBM+Histogram	SVM (RBF)	38 (22 CN, 16 AD)	LOOCV	94.5	91.5	96.6
12	2009	Gerardin et al. [119]	CN vs AD	sMRI (T1)	SPHARM	SVM (RBF)	46 (23 CN, 23 AD)	LOOCV	94	96	92
			CN vs MCI						83	83	84
13	2009	Chaves et al. [120]	CN vs AD	SPECT	VBM+NMSE	SVM (Linear)	79 (41 CN, 38 AD)	LOOCV	98.3	-	-
14	2010	Plant et al. [97]	CN vs AD	sMRI (T1)	VBM+Clustering	SVM (Linear)	74 (18 CN, 24 MCI, 32 AD)	LOOCV	90	96.88	77.78
			CN vs MCI						97.62	95.83	100
			MCI vs AD						95.83	88.89	100
15	2010	Segovia et al. [109]	CN vs AD	SPECT	GMM	SVM (Linear)	91 (41 CN, 50 AD)	LOOCV	94.5	-	-
				PET					96.67		
16	2010	Padilla et al. [123]	CN vs AD	SPECT	FDR+NMF	SVM (RBF)	97 (41 CN, 56 AD)	LOOCV	94.9	96.4	92.8
17	2011	Abdulkadir et al. [124]	CN vs AD	sMRI (T1)	VBM	SVM (Linear)	417 (226 CN, 191 AD)	LOOCV	87	-	-
18	2011	Illan et al. [149]	CN vs AD	SPECT	Image factorization	Ensemble SVM (RBF)	79 (41 CN, 38 AD)	LOOCV	96.91	94.64	100
19	2011	Cuingnet et al. [99]	CN vs AD	sMRI (T1)	VBM/ VolBM/ STAND- score	SVM (RBF)	509 (162 CN, 76 MCIc, 134 MCInc, 137 AD)	LOOCV	-	81	95
			CN vs MCIc						-	73	85
			MCInc vs MCIc						-	70	61
20	2012	Doecke et al. [77]	CN vs AD	Blood plasma	Statistical methods	SVM (Linear)	1131 (812 CN, 319 AD)	CV	-	80	85
21	2012	O'Dwyer et al. [100]	CN vs MCI	DTI	TBSS	SVM (RBF)	73 (40 CN, 19 naMCI, 14 aMCI)	10-fold	93	92.8	-
			CN vs aMCI vs naMCI						92.2	93.4	-
22	2013	Segovia et al. [127]	CN vs AD	SPECT	PLS	SVM (Linear)	97 (41 CN, 56 AD)	LOOCV	91.75	92.68	91.07

Table 2.1 (continued)

Sr. No.	Year	Authors	Target	Modality	Feature extraction	Machine learning	Dataset	Validation	Performance		
									Acc (%)	Sens (%)	Spec (%)
23	2013	Ramirez et al. [151]	CN vs AD	SPECT	FDR	SVM (RBF)	52 (23 CN, 29 AD)	LMOCV (M=5)	90.38	93.1	86.96
24	2013	Dukart et al. [107]	CN vs AD	FDG-PET+sMRI	VBM	SVM (Linear)	56 (28 CN, 28 AD)	LOOCV	100	100	100
25	2013	Ortiz et al. [115]	CN vs AD	sMRI (T1)	SOM clustering +FDR	SVM (RBF)	50 (25 CN, 25 AD)	LOOCV	92	96	98
26	2013	Haller et al. [114]	md-aMCI vs sd-aMCI	DTI	TBSS	SVM (RBF)	66 (18 sd-aMCI, 13 sd-fMCI, 35 md-aMCI)	10-fold	98.4	-	-
			md-aMCI vs sd-aMCI						97.7		
			sd-fMCI vs sd-aMCI						99.67		
27	2013	Lee et al. [112]	CN vs MCI	DTI	TBSS	SVM (RBF)	84 (39 CN, 45 MCI)	10-fold	100	100	100
28	2014	Hidalgo-Munoz et al. [129]	CN vs AD	sMRI (T1)	VBM +SVM-RFE	SVM (Linear)	370 (185 CN, 185 AD)	10-fold	100	-	-
29	2014	Lahmiri et al. [104]	CN vs MCI	sMRI (T2)	MSA	SVM (Polynomial)	33 (11 CN, 11 MCI, 11 AD)	10-fold	97.08	98.09	96.07
			MCI vs AD						97.5	100	94.93
30	2015	Khedher et al. [130]	CN vs AD	sMRI (T1)	VBM+Fast ICA	SVM (Linear)	818 (229 CN, 401 MCI, 188 AD)	2-fold	87.12	89.92	83.98
			CN vs MCI						77.62	80.27	74.49
			MCI vs AD						85.41	85.59	85.11
31	2015	Schmitter et al. [125]	CN vs AD	sMRI (T1)	VBM+VolBM	SVM (Linear)	818 (229 CN, 401 MCI, 188 AD)	LOOCV	-	86	91
			CN vs MCI							78	68
			MCI vs AD							69	67
			ADc vs ADnc							75	66
32	2015	Zhang et al. [116]	CN vs MCI vs AD	sMRI (T1)	VolBM+3D-DWT+PCA	SVM (RBF)	178 (97 CN, 57 MCI, 24 AD)	5-fold	81.5	-	-
33	2015	Zhang et al. [142]	CN vs AD	sMRI (T1)	DF+PCA	TWSVM	126 (98 CN, 28 AD)	10-fold	92.75	90.56	93.37
34	2015	Xu et al. [143]	CN vs AD	sMRI	Lasso features	S-LSTSVM (RBF)	329 (191 CN, 138 AD)	5-fold	92.1	92.52	92.07
35	2015	Ortiz et al. [122]	CN vs AD	FDG-PET+sMRI	VBM+SICE	SVM (Linear)	249 (68 CN, 111 MCI, 70 AD)	10-fold	92	96	86
			CN vs MCI						86	90	82
			MCI vs AD						84	87	81
36	2015	Retico et al. [128]	CN vs AD	sMRI (T1)	VBM+SVM-RFE	SVM (Linear)	635 (189 CN, 136 MCIc, 166 MCInc, 144 AD)	20-fold	88.9	-	-
			MCIc vs MCInc						70.7	-	-
37	2015	Zhang et al. [136]	CN vs AD	sMRI (T1)	PCA	SVM (Polynomial)	126 (98 CN, 28 AD)	10-fold	92.36	-	-
38	2016	Zhu et al. [102]	MCIc vs MCInc	sMRI	PIS	TS-SVM	151 (70 MCIc, 81 MCInc)	10-fold	82.5	-	-
39	2016	Moller et al. [152]	CN vs AD	sMRI (T1)	VBM	SVM (Linear)	178 (94 CN, 84 AD)	LOOCV	85	83	87
40	2016	Plocharsky et al. [126]	CN vs AD	sMRI (T1)	Morpholgical features (length, area, depth)	SVM (Linear)	210 (100 CN, 110 AD)	10-fold	87.9	90	86.7
41	2017	Alam et al. [86]	CN vs AD	sMRI (T1)	VolBM+KPCA	SVM (Multiple kernel)	293 (102 CN, 102 MCI, 89 AD)	10-fold	93.85	92.1	94.45
			CN vs MCI						86.54	84.85	87.74
			MCI vs AD						75.12	73.92	77.24

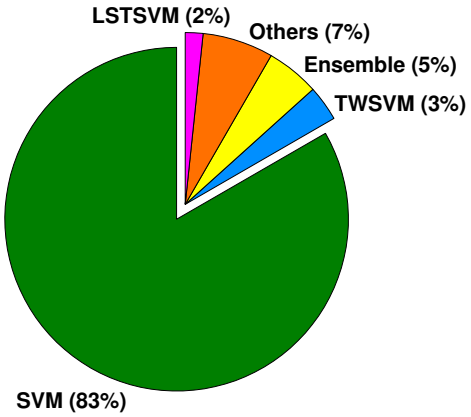
Table 2.1 (contd.)

Sr. No.	Year	Authors	Target	Modality	Feature extraction	Machine learning	Dataset	Validation	Performance		
									Acc (%)	Sens (%)	Spec (%)
42	2017	Khedher et al. [131]	CN vs AD	sMRI (T1)	ICA	SVM (RBF)	818 (229 CN, 401 MCI, 188 AD)	k-fold	89	92	86
			CN vs MCI						79	82	76
			MCI vs AD						85	85	86
43	2017	Beheshti et al. [98]	CN vs AD	sMRI (T1)	VBM+GA	SVM (Linear)	458 (162 CN, 65 sMCI, 71 pMCI, 160 AD)	10-fold	93.01	89.13	96.8
			pMCI vs sMCI						75	76.92	73.23
44	2017	Long et al. [153]	CN vs AD	sMRI (T1)	MDS+PCA	SVM (Linear)	427 (135 CN, 132 sMCI, 95 pMCI, 65 AD)	10-fold	96.5	93.85	97.78
			CN vs pMCI						97.1	87.37	94.82
			sMCI vs pMCI						88.99	86.32	90.91
45	2017	Tangaro et al. [87]	CN vs AD	sMRI (T1)	VolBM	SVM (Linear)	372 (117 CN, 86 MCIc, 71 MCInc, 98 AD)	10-fold	100	-	-
			MCIc vs MCInc						83.4	-	-
46	2017	Lu et al. [144]	CN vs MCI	FDG-PET	VBM	RF-RSVM	272 (152 CN, 120 MCI)	3-fold	90.53	90.63	93.33
47	2017	Alam et al. [145]	CN vs AD	sMRI	DTCWT/LDA	TWSVM	237 (130 CN, 137 AD)	10-fold	96.88	97.72	95.61
48	2017	Hojjati et al. [110]	MCIc-MCInc	rs-fMRI	PCC+F-score	SVM (Linear)	80 (18 MCIc, 62 MCInc)	9-fold	91.4	83.24	90.1
49	2017	Kulkarni et al. [132]	CN vs AD	EEG	ICA/ Wavelet/ Spectral	SVM	100 (50 CN, 50 AD)	LOOCV	96	-	-
50	2018	Sun et al. [146]	CN vs AD	sMRI (T1)	VBM+PCC	Group lasso SVM	509 (162 CN, 134 sMCI, 76 pMCI, 137 AD)	5-fold	95.1	93.8	83.8
			CN vs MCI						70.8	72.1	69.1
			sMCI vs pMCI						65.4	67.6	64.2
			MCI vs AD						65.7	63.2	67.3
51	2018	Zeng et al. [147]	CN vs AD	sMRI (T1)	PCA+PSO and SDPSO	SVM (RBF)	361 (92 CN, 82 sMCI, 95 pMCI, 92 AD)	10-fold	82.5	-	-
			sMCI vs pMCI						69.23		
			CN vs sMCI						76.92		
			CN vs pMCI						85.71		
			sMCI vs AD						72.94		
			pMCI vs AD						57.14		
52	2018	Lahmiri et al. [137]	CN vs AD	sMRI (T1)	VolBM	SVM (Polynomial)	70 (35 CN, 35 AD)	10-fold	100	100	100
53	2018	Kamathe et al. [105]	CN vs AD	sMRI (T1, T2)+PD	ICA	SVM (Polynomial)	20 (15 CN, 5 AD)	-	100	-	-
54	2018	Bi et al. [148]	CN vs AD	rs-fMRI	PCC	RSVM (RBF)	61 (36 CN, 25 AD)	-	94.44	-	-
55	2018	Mazaheri et al. [133]	MCIc vs (CN + MCInc)	EEG	TFRs	SVM (RBF)	36 (11 CN, 10 MCInc, 15 MCIc)	LOOCV	-	80	95
56	2018	Paraskevaidi et al. [78]	CN vs AD	Blood plasma	PCA-LDA	SVM	41 (15 CN, 11 eAD, 15 lAD)	LOOCV	-	84	86
			CN vs lAD							84	77
			eAD vs lAD							66	83
57	2018	Zhang et al. [113]	CN vs AD	DTI	LDH +SVM-RFE	SVM (Linear)	213 (51 CN, 75 eMCI, 39 lMCI 48 AD)	LOOCV	89.9	-	-
			CN vs eMCI						88.1		
			CN vs lMCI						100		
			eMCI vs lMCI,						92.98		
			eMCI vs AD						84.55		
			lMCI vs AD						97.7		

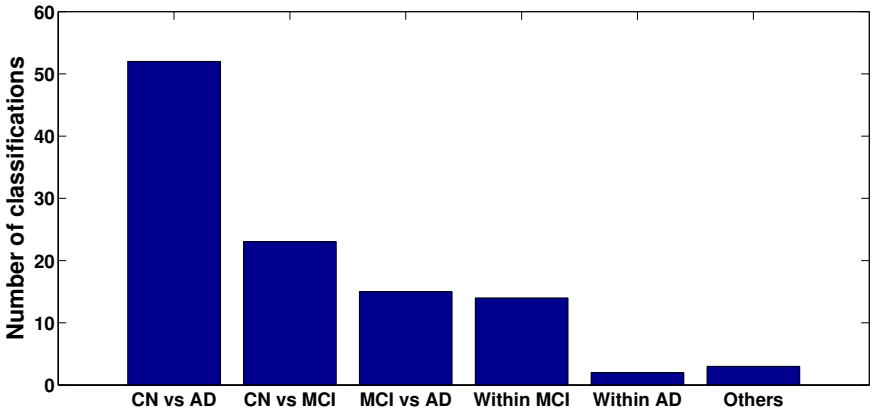
Table 2.1 (contd.)

Sr. No.	Year	Authors	Target	Modality	Feature extraction	Machine learning	Dataset	Validation	Performance		
									Acc (%)	Sens (%)	Spec (%)
58	2019	Peng et al. [138]	CN vs AD	sMRI+PET +SNP	Volume + Mean intensity features	SVM (Multiple kernel)	189 (47 CN, 93 MCI, 49 AD)	10-fold	96.1	97.3	94.9
			CN vs MCI						80.3	85.6	69.8
			MCI vs AD						76.9	65.9	82.7
59	2019	Sheng et al. [111]	CN vs eMCI	fMRI	RF-score	SVM	96 (24 CN, 24 eMCI, 24 lMCI, 24 AD)	5-fold	93.8	-	-
			CN vs LMCI						95.8		
			CN vs AD						95.8		
			eMCI vs lMCI						87.5		
			lMCI vs AD						91.7		
60	2019	Gosztolya et al. [134]	CN vs MCI	Acoustic signal	MFCC	SVM (Linear)	75 (25 CN, 25 MCI, 25 mAD)	5-fold	80	-	75.9
			CN vs mAD						86	-	87.5
			MCI vs mAD						80	-	85.7

Abbreviations: **LMOCV**- Leave M out cross validation, **CV**- Cross validated, **CT**- Classification tree, **ODC**- Optimally differentiating clusters, **SPHARM**- Spherical harmonics coefficients, **VolBM**- Volume based morphometry, **eMCI**- Early MCI, **lMCI**- Late MCI, **mAD**- Mild AD, **ADc**- AD converter, **ADnc**- AD non-converter, **MDS**- Multi dimensional scaling, **PCC**- Pearson correlation coefficient, **SNP**- Single nucleotide polymorphisms, **MFCC**- Mel frequency cepstral coefficients, **DF**- Displacement field, **MSA**- Multiscale analysis, **S-LSTSVM**-Structural least squares twin support vector machine, **RSVM**- Random SVM, **rs-fMRI**- Resting state fMRI, **PIS**- Partial image sequence, **TS-SVM**- Temporally structured SVM, **RF-score**- Relief feature score, **STAND**- score- Structural abnormality index score, **sd-aMCI**- Single domain amnestic MCI, **md-aMCI**- Multiple domain amnestic MCI, **sd-fMCI**- Single domain frontal MCI, **TBSS**- Tract-based spatial statistics, **TFR**- Time frequency representation, **KPCA**- Kernel PCA, **LDH**- Local diffusion homogeneity.



(a)



(b)

Figure 2.5: Plot showing usage of (a) different variants of SVM and (b) different target groups for AD.

be large for training of the model. Further, the data collection for Alzheimer's includes noise from various sources. Therefore, noise insensitive techniques must be applied for AD classification.

In the following section, we discuss a classification algorithm using projection based approach, and an unsupervised SVM based algorithm for clustering problems.

2.5 Projection based twin SVM and clustering algorithms

A novel approach based on projection axes rather than classifying hyperplanes is proposed as projection twin support vector machine [155], and extended for regression problems in [156,157]. To improve the computation cost, Shao et al. [158] proposed an efficient least squares projection twin support vector machine (LSPTSVM). LSPTSVM has been used for various classification problems [159, 160]. However, the algorithms discussed in the previous sections are supervised learning algorithms, needing information about true labels of data samples in the training process.

For unsupervised learning i.e., where the labels of training data are unknown, algorithms like k-means clustering [161], and fuzzy c-means (FCM) [162] clustering are proposed in the past. In FCM, clustering is performed based on distance from cluster centres with fuzzy membership value for each cluster. However, plane based clustering algorithms are also proposed, such as the k-plane clustering (kPC) algorithm [163]. In kPC, a plane is constructed for each cluster by solving an eigenvalue problem. Some other plane based clustering algorithms are proposed in [164, 165]. In 2015, Wang et al. [166] proposed an unsupervised algorithm termed as twin support vector clustering (TWSVC), improving the proximal plane clustering algorithm [164]. To include regularization in TWSVC, a twin bounded support vector clustering (TBSVC) is proposed [167], leading to improved generalization performance. In order to reduce the computation cost of TWSVC, least squares twin support vector clustering (LSTWSVC) is formulated in [168]. In LSTWSVC, a set of linear equations is solved instead of QPPs. A fuzzy least squares twin support vector clustering (FLSTWSVC) [168] is also proposed by including fuzzy membership values for the data points.

In the next subsections, we briefly discuss the formulation of a classification algorithm i.e., LSPTSVM [158], and a clustering algorithm i.e., TWSVC [166].

2.5.1 Least squares projection twin support vector machine (LSPTSVM)

Linear LSPTSVM [158] generates two non-parallel hyperplanes based on the following optimization problems,

$$\begin{aligned} \min_{w_1} \ & \frac{1}{2} w_1^T S_1 w_1 + \frac{c_1}{2} \sum_{q=1}^{m_2} (\xi_q)^2 + \frac{c_3}{2} \|w_1\|^2 \\ s.t. \ & w_1^T x_q^{(2)} - w_1^T \frac{1}{m_1} \sum_{p=1}^{m_1} x_p^{(1)} + \xi_q = 1, \quad q = 0, 1, \ldots, m_2, \end{aligned} \tag{2.38}$$

$$\begin{aligned} \min_{w_2} \ & \frac{1}{2} w_2^T S_2 w_2 + \frac{c_2}{2} \sum_{p=1}^{m_1} (\eta_p)^2 + \frac{c_4}{2} \|w_2\|^2 \\ s.t. \ & -\left(w_2^T x_p^{(1)} - w_2^T \frac{1}{m_2} \sum_{q=1}^{m_2} x_q^{(2)} \right) + \eta_p = 1, \quad p = 0, 1, \ldots, m_1, \end{aligned} \tag{2.39}$$

where $c_i, i = 1, \ldots, 4$ are positive parameters, and ξ, η are slack variables. The matrices S_1 and S_2 are written as

$$S_1 = \sum_{p=1}^{m_1} \left(x_p^{(1)} - \frac{1}{m_1} \sum_{p=1}^{m_1} x_p^{(1)} \right) \left(x_p^{(1)} - \frac{1}{m_1} \sum_{p=1}^{m_1} x_p^{(1)} \right)^T, \tag{2.40}$$

$$S_2 = \sum_{q=1}^{m_2} \left(x_q^{(2)} - \frac{1}{m_2} \sum_{q=1}^{m_2} x_q^{(2)} \right) \left(x_q^{(2)} - \frac{1}{m_2} \sum_{q=1}^{m_2} x_q^{(2)} \right)^T. \tag{2.41}$$

Now, QPP (2.38) can be written using matrices of data points in the objective function [158],

$$L = \frac{1}{2} w_1^T S_1 w_1 + \frac{c_1}{2} \left\| -X_2 w_1 + \frac{1}{m_1} e_2 e_1^T X_1 w_1 + e_2 \right\|^2 + \frac{c_3}{2} \|w_1\|^2, \tag{2.42}$$

where e_1, e_2 are vectors of ones of appropriate dimensions.

Setting the gradient of Eq. (2.42) w.r.t. w_1 equal to 0 and solving, we get

$$w_1 = \left(\frac{S_1}{c_1} + \Big(-X_2 + \frac{1}{m_1} e_2 e_1^T X_1 \Big)^T \Big(-X_2 + \frac{1}{m_1} e_2 e_1^T X_1 \Big) + \frac{c_3}{c_1} I \right)^{-1} \Big(X_2 - \frac{1}{m_1} e_2 e_1^T X_1 \Big)^T e_2, \tag{2.43}$$

where I is identity matrix of appropriate dimension.

Similarly, w_2 is calculated as

$$w_2 = - \left(\frac{S_2}{c_2} + \Big(X_1 - \frac{1}{m_2} e_1 e_2^T X_2 \Big)^T \Big(X_1 - \frac{1}{m_2} e_1 e_2^T X_2 \Big) + \frac{c_4}{c_2} I \right)^{-1} \Big(X_1 - \frac{1}{m_2} e_1 e_2^T X_2 \Big)^T e_1. \tag{2.44}$$

For a testing sample x_t, the class is determined as follows,

$$class\ (x_t) = arg \min_{i=1,2} \left| w_i^T x_t - w_i^T \frac{1}{m_i} \sum_{k=1}^{m_i} x_k^{(i)} \right|. \tag{2.45}$$

2.5.2 Twin support vector clustering (TWSVC)

TWSVC [166] generates non-parallel clustering hyperplanes by solving the following optimization problem:

$$\begin{aligned} \min_{w_i^{j+1}, b_i^{j+1}, \xi_i^{j+1}} \quad & \frac{1}{2} \| X_i w_i^{j+1} + b_i^{j+1} e \|^2 + c_1 e^T \xi_i^{j+1} \\ s.t. \quad & T(|\overline{X}_i w_i^{j+1} + b_i^{j+1} e|) \geq e - \xi_i^{j+1},\ \xi_i^{j+1} \geq 0, \\ & i = 0, 1, \ldots, N, \end{aligned} \tag{2.46}$$

where $c_1 > 0$ is the penalty parameter, T(.) is the Taylor series expansion, and ξ_i^{j+1} is the slack variable, $j = 0, 1, \ldots,$ and e is vector of ones of appropriate dimension.

By using the subgradient [169] of $|\overline{X}_i w_i^j + b_i^j e|$ w.r.t. w_i^j and b_i^j and the Taylor series expansion [166, 168], we get

$$\min_{w_i^{j+1}, b_i^{j+1}, \xi_i^{j+1}} \ \frac{1}{2}\|X_i w_i^{j+1} + b_i^{j+1} e_i\|^2 + c_1 e^T \xi_i^{j+1}$$
$$s.t. \qquad diag(sign(\overline{X}_i w_i^j + b_i^j e))(\overline{X}_i w_i^{j+1} + b_i^{j+1} e) \geq e - \xi_i^{j+1},\ \xi_i^{j+1} \geq 0. \quad (2.47)$$

The dual problem of QPP (2.46) is written as

$$\min_{\lambda} \ \frac{1}{2}\lambda^T B(A^T A)^{-1} B^T \lambda - e^T \lambda$$
$$s.t.\ 0 \leq \lambda \leq c_1 e, \quad (2.48)$$

where $B = diag\big(sign(\overline{X}_i w_i^j + b_i^j e)\big)\big[\overline{X}_i \quad e\big], A = [X_i \quad e]$, and λ is the vector of Lagrange multipliers.

The hyperplane for each cluster is found using the following equation:

$$[w_i^{j+1} \quad b_i^{j+1}]^T = (A^T A)^{-1} B^T \lambda, \quad i = 0, 1, \ldots, N. \quad (2.49)$$

2.6 Research methodology

In the section, we present the details about the datasets, experimental setup, and various performance metrics and other formulae used in this thesis.

2.6.1 Datasets

The real world datasets are downloaded from UCI [170], and KEEL repository [171]. For Alzheimer's disease, all MRI images used in this work were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (`adni.loni.usc.edu`). ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The main goal of ADNI is to find out the effectiveness of neuroimaging techniques like MRI, positron emission tomography (PET), other biological markers, and clinical neuropsychological tests to estimate the onset of

Alzheimer's disease from the state of mild cognitive impairment. For more information, visit `www.adni-info.org`.

2.6.2 Experimental setup

All the experiments are performed on a PC running on 64 bit Windows 10 operating system, 3.60 GHz Intel® core™ i7-7700 processor, 16 GB of RAM under MATLAB R2008b environment. 5-fold cross validation is used for parameter selection in all the methods. MOSEK optimization toolbox (`http://www.mosek.com`) is used to solve the QPPs. For non-linear case, radial basis function (RBF) or Gaussian kernel is used in all the algorithms, which is defined as

$$K(x, y) = exp\left(\frac{-1}{2\mu^2}\|x - y\|^2\right), \tag{2.50}$$

where x and y are vectors, and μ is a scalar parameter.

The imbalance ratio (IR) is calculated as

$$IR = \frac{\text{Number of majority class samples}}{\text{Number of minority class samples}}. \tag{2.51}$$

In the works presented in this thesis, we use the terms positive and negative class for minority and majority class respectively.

2.6.3 Performance metrics

The different performance metrics used in this thesis are as follows:

(i). Accuracy = $\frac{\text{TP+FP}}{\text{TP+TN+FP+FN}}$, where TP=True positive, TN=True negative, FP=False positive, and FN=False negative.

(ii). Sensitivity = $\frac{\text{TP}}{\text{TP+FP}}$

(iii). Specificity = $\frac{\text{TN}}{\text{TP+FP}}$

(iv). F1 score = $\frac{2(\text{TP})}{2(\text{TP})+\text{FP+FN}}$

(v). For class imbalanced data, the accuracy is calculated using area under receiver operating characteristics (ROC) curve i.e. AUC [55] is used which is defined as:

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2}, \tag{2.52}$$

where TP_{rate} is the true positive classification of minority (positive) class, and FP_{rate} is the false positive classification of majority (negative) class data points.

(vi). The clustering accuracy for l data samples with y labels is measured using the following similarity matrix $L \in \mathbb{R}^{l \times l}$ [166],

$$L(i,j) = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{otherwise.} \end{cases}$$

Now, let L_p is similarity matrix of predicted cluster labels, and L_a is the similarity matrix of actual labels. Then, the accuracy is defined as the rand index [166],

$$Accuracy = \frac{n_0 + n_1 - l}{l^2 - l} \times 100\%, \tag{2.53}$$

where n_0 is the number of zeros in L_a and L_p, and n_1 is number of ones in L_a and L_p.

2.6.4 Statistical tests

To check the statistical significance of the proposed algorithms in this thesis, we used Friedman test with the corresponding post-hoc test [172] using the average ranks of the algorithms based on accuracy of the datasets. First, we assume that all the methods are equivalent under null hypothesis. The Friedman statistic is computed using the χ_F^2 value as follows:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_{i=1}^{k} R_i^2 - \frac{k(k+1)^2}{4} \right], \tag{2.54}$$

where R_i is the average rank on N datasets for i^{th} method, k denotes total number of methods.

Then, the F_F value is calculated using the following formula:

$$F_F = \frac{(N-1)(\chi_F^2)}{N \times (k-1) - \chi_F^2}. \tag{2.55}$$

The F_F is distributed as the F-distribution with $\big(k-1, (N-1)(k-1)\big)$ degrees of freedom. For significant difference between the methods at α level of significance, the value of F_F must be more than the critical value.

To check the pairwise difference between the proposed and existing algorithms, we use the Nemenyi posthoc test. The critical difference is calculated using the following:

$$CD = t_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{2.56}$$

where t_α is the critical value for α level of significance, and CD is the critical difference for k algorithms and N datasets. For significant pairwise difference between the methods at significance level α, the difference in the average ranks of the methods should be atleast the CD.

This chapter presented a review on the works related to algorithms proposed in this thesis. The following chapter presents two novel algorithms to deal with the problem of class imbalanced data with SVM.

Chapter 3

Twin support vector machine for class imbalance learning

In this chapter, we present two efficient twin SVM based algorithms to reduce the effect of class imbalance on the classification of data. Section 3.1 presents the robust fuzzy least squares twin support vector machine for class imbalance learning (RFLSTSVM-CIL) algorithm[1]. The RFLSTSVM-CIL algorithm utilizes a novel fuzzy membership function proposed in this work.

To include prior information about data distribution in the classification of imbalanced data, the idea of universum is incorporated in the proposed reduced universum twin support vector machine for class imbalance learning (RUTSVM-CIL)[2]. As per our survey, the concept of universum is used for the first time to solve class imbalance problem. Section 3.2 discusses the RUTSVM-CIL algorithm. First, we present the RFLSTSVM-CIL algorithm in the following section.

[1] **B. Richhariya**, M. Tanveer. A robust fuzzy least squares twin support vector machine for class imbalance learning. *Applied Soft Computing*, Elsevier, 71: 418-432, 2018, DOI: `https://doi.org/10.1016/j.asoc.2018.07.003`.
[**SCI Indexed Impact Factor: 6.725**]

[2] **B. Richhariya**, M. Tanveer. A reduced universum twin support vector machine for class imbalance learning. *Pattern Recognition*, Elsevier, 102:107150, 2020, DOI: `https://doi.org/10.1016/j.patcog.2019.107150`.
[**SCI Indexed Impact Factor: 7.740**]

3.1 A robust fuzzy least squares twin support vector machine for class imbalance learning (RFLSTSVM-CIL)

In most of the applications on classification, there is imbalance in the number of samples of the classes, which leads to incorrect classification of the data points of the minority class. Further, while dealing with imbalanced data, noise poses a major challenge in various applications. To resolve these problems, in this work we propose a robust fuzzy least squares twin support vector machine for class imbalance learning, termed as RFLSTSVM-CIL using 2-norm of the slack variables which makes the optimization problem strongly convex. In order to reduce the effect of outliers, we propose a novel fuzzy membership function specifically for class imbalance problems. Our proposed function gives appropriate weights to the datasets and also incorporates the knowledge about the imbalance ratio of data. In our proposed model, a pair of system of linear equations is solved instead of solving a quadratic programming problem (QPP), which makes our model efficient in terms of computation time. To check the performance of our proposed approach, several numerical experiments are performed on synthetic and real world benchmark datasets. The proposed RFLSTSVM-CIL model has shown better generalization performance in comparison to existing methods in terms of AUC and training time.

To give appropriate membership to the majority class, we propose a new fuzzy membership function for imbalance datasets. In the previous work on fuzzy membership for imbalance data, the range of fuzzy membership is fixed for datasets with different imbalance ratios. To overcome this drawback, our function uses information about the imbalance ratio (IR) and gives appropriate range of the fuzzy membership to different datasets. Moreover, we present a novel 2-norm based robust fuzzy least squares twin support vector machine for class imbalance learning (RFLSTSVM-CIL). To justify the effectiveness of our proposed approach, several numerical experiments are performed on synthetic and real world benchmark datasets.

3.1.1 Proposed fuzzy membership function

There are some drawbacks in the existing fuzzy membership functions used with SVM. For example, the centroid based fuzzy membership function only gives membership values on the basis of the distance from its centroid, but does not take into account whether a data point is near to the region of its own class or the other. Similarly, in other fuzzy functions used for class imbalance problems [55], this idea of proximity to its class centroid with information about the other class is not considered. Also, the information about the extent of imbalance in the data is not utilised in the previous works. Therefore, the proposed approach includes the information about the imbalance ratio (IR) which control the range of the membership values. Motivated by the works of [173, 174], we propose a new fuzzy membership function for class imbalance problem.

For negative class, the fuzzy membership function is as follows:

$$mem = \Big(\frac{1}{1+IR}\Big) + \Big(\frac{IR}{1+IR}\Big)\Bigg(\frac{exp\big(c_0((d_1-d_2)/d - d_2/r_2)\big) - exp\big(-2c_0\big)}{exp(c_0) - exp(-2c_0)}\Bigg), \tag{3.1}$$

where IR is the imbalance ratio, d_1 is Euclidean distance from centroid of positive class, and d_2 is Euclidean distance from centroid of negative class, d is the distance between the centroid of the binary classes, r_2 is the maximum distance of the data points of negative class from its centroid, and c_0 decides the scale of the exponential function. The membership is assigned as 1 to all the data points of the positive class which is having the lesser number of samples [55].

The proposed fuzzy membership function is based on the following aspects of the data points.

(i). Proximity of the majority class data point to the centroid of the other classes.

(ii). Proximity of the majority class data points to their own class.

Properties of the proposed function:

(i). The membership value of the negative class ranges from $\left(\frac{1}{1+IR}\right)$ to 1 based on the position of the data points w.r.t. centroids of the two classes. The range of the membership value depends on the imbalance ratio (IR).

(ii). The membership value for the negative data point based on its proximity to the positive class depends on the variable $(d_1 - d_2)/d$ in the membership function.

(iii). The penalty for the outliers which are proximal to the negative class is taken care by d_2/r_2.

(iv). The membership value is equal to 1 when $d_2 = 0$ which makes $d_1 = d$.

(v). The membership value of data point is equal to $\left(\frac{1}{1+IR}\right)$ when it is closest to the positive class centroid i.e., $d_1 = 0$ resulting in $d = r_2$, and farthest from the centroid of the negative class i.e., $d_2 = r_2$.

(vi). One can observe from Fig. 3.1 that if the outlier data point of the majority class (negative class) is in the positive class region, then the penalty on the membership value is higher as compared to when it is on its own side i.e. negative class.

In class imbalance problems, the objective is to classify the data points of the minority class more effectively. For achieving this property in an effective manner, the proposed fuzzy membership function gives the membership according to the following cases, as illustrated in Fig. 3.1 for an artificial binary dataset:

Case 1 Negative data point closer to the centroid of its own class: **High membership**.

Case 2 Negative data point away from its own centroid but relatively closer to its own centroid as compared to the other class: **Low membership**.

Case 3 Negative data point away from its own centroid and relatively closer to the positive class centroid: **Very low membership**.

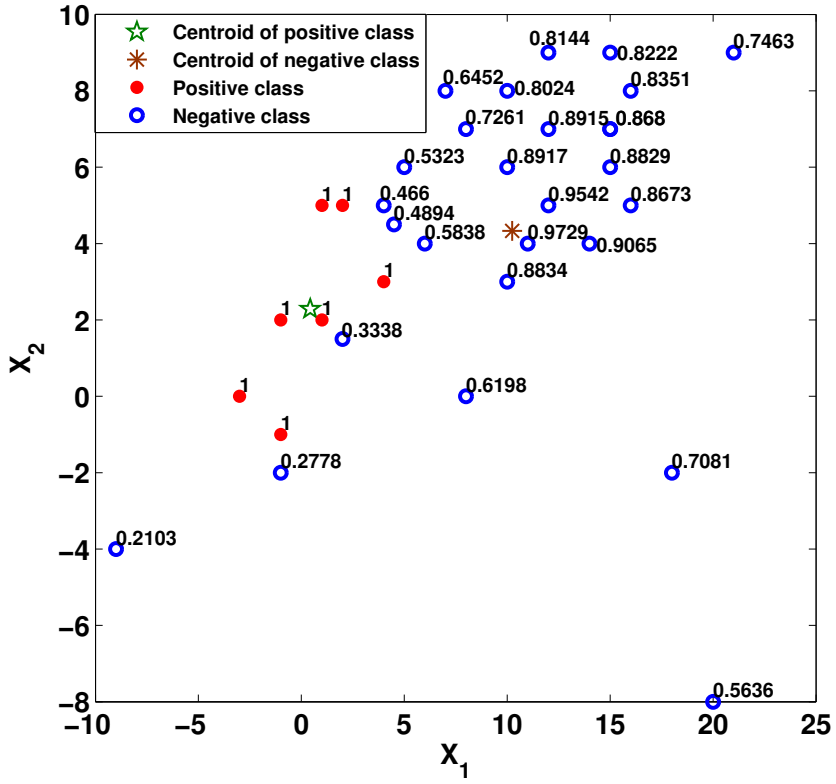


Figure 3.1: Plot of artificial dataset ($IR = 3.57$) showing membership values of the data points based on the proposed membership function for $c_0 = 0.5$.

3.1.2 Linear RFLSTSVM-CIL

In the proposed approach, the 2-norm of the weighted slack vector is used for giving fuzzy membership values to data points in the constraints of the optimization problem. The optimization problems of linear RFLSTSVM-CIL are written as

$$\begin{aligned} \min_{w_1, b_1, \xi} \quad & \frac{1}{2}\|X_1 w_1 + e_1 b_1\|^2 + \frac{c_1}{2}\|S_2 \xi\|^2 \\ s.t. \quad & -(X_2 w_1 + e_2 b_1) + \xi = e_2, \end{aligned} \tag{3.2}$$

$$\begin{aligned} \min_{w_2, b_2, \eta} \quad & \frac{1}{2}\|X_2 w_2 + e_2 b_2\|^2 + \frac{c_2}{2}\|S_1 \eta\|^2 \\ s.t. \quad & (X_1 w_2 + e_1 b_2) + \eta = e_1, \end{aligned} \tag{3.3}$$

where X_1 and X_2 are matrices of class 1 (minority) and class 2 (majority) containing p and q number of samples respectively. S_1 is identity matrix of dimension p, and S_2 is diagonal matrix of dimension q containing the fuzzy membership values at the diagonal places. The slack variables are represented by ξ, η with $c_1, c_2 > 0$ as the penalty parameters and e_1, e_2 are the vectors of ones of appropriate dimension.

Using the equality constraints of QPP (3.2) in its objective function, the QPP is written as

$$\min_{w_1,\, b_1} \quad \frac{1}{2}\|X_1 w_1 + e_1 b_1\|^2 + \frac{c_1}{2}\|S_2(X_2 w_1 + e_2 b_1 + e_2)\|^2. \tag{3.4}$$

Taking the gradient of QPP (3.4) with respect to w_1 and b_1 and equating to 0, we get

$$X_1^T(X_1 w_1 + e_1 b_1) + c_1(S_2 X_2)^T(S_2(X_2 w_1 + e_2 b_1 + e_2)) = 0, \tag{3.5}$$

$$e_1^T(X_1 w_1 + e_1 b_1) + c_1(S_2 e_2)^T(S_2(X_2 w_1 + e_2 b_1 + e_2)) = 0. \tag{3.6}$$

Combining equations (3.5) and (3.6), and solving [18], we get,

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -\left(T^T T + \frac{1}{c_1} R^T R\right)^{-1} T^T S_2 e_2, \tag{3.7}$$

where $R = [X_1 \;\; e_1]$, and $T = [S_2 X_2 \;\; S_2 e_2]$.

Similarly, the other hyperplane is computed by the following,

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \left(R^T R + \frac{1}{c_2} T^T T\right)^{-1} R^T S_1 e_1, \tag{3.8}$$

where $R = [S_1 X_1 \;\; S_1 e_1]$, and $T = [X_2 \;\; e_2]$.

For reducing the computation time of finding the inverse, Sherman-Morrison-Woodbury (SMW) formula [40] is used for the equations (3.7) and (3.8) and inverses

of smaller dimensions are solved. For a new data point x, the class assignment is done using Eq. (2.11).

If we give the membership values for both the classes according to our fuzzy membership function then our proposed RFLSTSVM-CIL can also be applied to datasets with no class imbalance. Further, if the membership values of both the classes are set to 1 then our proposed RFLSTSVM-CIL reduces to the standard LSTSVM. So, we can say that LSTSVM is a special case of RFLSTSVM-CIL.

3.1.3 Non-linear RFLSTSVM-CIL

The formulation of the non-linear RFLSTSVM-CIL is written as

$$\begin{aligned} \min_{w_1, b_1, \xi} \quad & \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + \frac{c_1}{2}\|S_2\xi\|^2 \\ s.t. \quad & -(K(X_2, D^T)w_1 + e_2 b_1) + \xi = e_2, \end{aligned} \tag{3.9}$$

$$\begin{aligned} \min_{w_2, b_2, \eta} \quad & \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + \frac{c_2}{2}\|S_1\eta\|^2 \\ s.t. \quad & (K(X_1, D^T)w_2 + e_1 b_1) + \eta = e_1, \end{aligned} \tag{3.10}$$

where matrix $D = [X_1^T \ X_2^T]^T$, $K(X_1, D^T), K(X_2, D^T)$ are the kernel matrices of class 1 and 2 respectively.

Using the constraints of (3.9) in its objective function, the QPP is written as

$$\min_{w_1, b_1} \quad \frac{1}{2}\|K(X_1, D^T)w_1 + eb_1\|^2 + \frac{c_1}{2}\|S_2(K(X_2, D^T)w_1 + e_2 b_1 + e_2)\|^2. \tag{3.11}$$

Taking the gradient of (3.11) with respect to w_1 and b_1 and equating to 0, we get

$$\begin{aligned} & K(X_1, D^T)^T(K(X_1, D^T)w_1 + e_1 b_1) \\ & + c_1(S_2 K(X_2, D^T))^T(S_2(K(X_2, D^T)w_1 + e_2 b_1 + e_2)) = 0, \end{aligned} \tag{3.12}$$

$$e_1^T(K(X_1, D^T)w_1 + e_1b_1) + c_1(S_2e_2)^T(S_2(K(X_2, D^T)w_1 + e_2b_1 + e_2)) = 0. \quad (3.13)$$

Similarly as in the linear case, one can write in the following form,

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -\left(T^TT + \frac{1}{c_1}R^TR\right)^{-1} T^TS_2e_2, \quad (3.14)$$

where $R = [K(X_1, D^T)\ \ e_1]$ and $T = [S_2K(X_2, D^T)\ \ S_2e_2]$.

Similarly, for the other hyperplane the parameters are computed as

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \left(R^TR + \frac{1}{c_2}T^TT\right)^{-1} R^TS_1e_1, \quad (3.15)$$

where $R = [S_1K(X_1, D^T)\ \ S_1e_1]$ and $T = [K(X_2, D^T)\ \ e_2]$.

For reducing the computation time of finding the inverse, SMW formula [40] is used for the equations (3.14) and (3.15) and inverses of smaller dimensions are solved. The class of a new data sample $x \in R^n$ is predicted based on the perpendicular distances from the hyperplanes $K(x^T, D^T)w_1 + b_1$ and $K(x^T, D^T)w_2 + b_2$ and the class label of the nearer hyperplane is assigned to it.

3.1.4 Computational complexity

Our proposed approach of RFLSTSVM-CIL incorporates the 2-norm of the slack variable with fuzzy membership values in the formulation of LSTSVM. Similar to LSTSVM, our proposed RFLSTSVM-CIL solves two systems of linear equations which involve the inversion of matrices.

In the formulation of LSTSVM, the calculation of two inverses of size $(m + 1)$ is required where $m = p + q$, p and q are number of data points of positive and negative class. So, reduce the computation of the inverses the Sherman–Morrison–Woodbury (SMW) formula [40] is used, where three inverse of smaller sizes are solved. In case of our proposed algorithm the size of the invertible matrices are same as in LSTSVM, so there is no computation overhead in terms of solving the optimization problem as

compared to LSTSVM. Further, in comparison to SVM and TWSVM our proposed RFLSTSVM-CIL is computationally efficient as it is calculating the solution of linear equations as in LSTSVM.

The additional computation involved in our proposed method is the calculation of the fuzzy membership function. In comparison to the existing fuzzy based approaches, our fuzzy membership function is efficient in terms of computation cost. The traditional fuzzy based functions have the time complexity of $O(m)$. This is because these functions calculate the fuzzy membership of all the data points based on measures like distance from centroid with linear and exponential decay functions. Our proposed fuzzy based function has the time complexity as $O(q)$ where $q < m$ and q is the number of samples of the negative class. This is due to the fact that our proposed function calculates fuzzy values only for the majority class and assigns the membership value as 1 for the minority class.

3.1.5 Experimental results

The performance of the proposed methods is compared with several existing algorithms on various synthetic and real world imbalanced datasets. EFSVM [55], TWSVM [12], FTWSVM$_{lin}$ & FTWSVM$_{exp}$ [42, 175, 176], Universum Twin Support Vector Machine (UTSVM) [27], and LSTSVM [18] are compared with the proposed method RFLSTSVM-CIL in terms of accuracy and training time. UTSVM incorporates the notion of prior information about the data. So we have compared our proposed RFLSTSVM-CIL with UTSVM in case of imbalanced data. To show the effectiveness of our proposed fuzzy membership function, we show the comparison of our proposed RFLSTSVM-CIL with the novel fuzzy function to the proposed algorithm using existing fuzzy functions for assigning the weights. The centroid based fuzzy membership functions with linear (FLSTSVM-CIL$_{lin}$) and exponential decay (FLSTSVM-CIL$_{exp}$) are also the proposed algorithms using the existing fuzzy membership functions.

The AUC is calculated as mean AUC with standard deviation for five iterations on the testing data. In each iteration, one part is used for testing and the remaining

data for training. The time is calculated in seconds and averaged over five iterations. The value of the penalty parameter is taken as $c = c_1 = c_2 = c_u$ from the set $\{10^{-5}, 10^{-4}, ..., 10^{5}\}$, where c_u is used in UTSVM [27] and μ is taken from the set $\{2^{-5}, 2^{-4}, ..., 2^{5}\}$ for all the cases. For RFLSTSVM-CIL, c_0 is chosen from the set $\{0.5, 1, 1.5, 2, 2.5\}$. For EFSVM [55], β is considered as 0.05, l is taken as 10, and the value of k is chosen from $\{3, 5, 7, 9, 11\}$. For FTWSVM$_{lin}$, δ is taken as 0.01 and for FTWSVM$_{exp}$, β is chosen from the set $\{0.1, 0.3, 0.5, 0.7, 1\}$. In UTSVM, ϵ is taken from the set $\{0.1, 0.3, 0.5, 0.6\}$, number of universum samples i.e. u is taken as 10% of the training data, and random averaging scheme [27] is used for the generation of universum. The AUC is calculated in terms of percentage for all the algorithms.

3.1.5.1 Synthetic datasets

To analyse the performance of our proposed method, we performed experiments on different synthetic datasets. We used 6 synthetic datasets to test the performance of our proposed approach. The datasets containing noise are taken from KEEL imbalanced dataset repository [171,177] having 2 classes where the data points are randomly and uniformly distributed in the two-dimensional space (both attributes are real valued). The noisy datasets are namely 04clover5z-600-5-60-BI, 03subcl5-600-5-30-BI, 03subcl5-600-5-50-BI and 03subcl5-600-5-60-BI with the disturbance ratio as 60%, 30%, 50% and 60% respectively [177].

We also performed experiments on Crossplane (XOR) dataset [178] generated with different number of samples and imbalance ratios as shown in Table 3.1. For the generation of the datasets, randomized values of data points are used in the equation of a line i.e., $y = kx + b$ to generate the dataset. The parameters for slope and intercept i.e., k and b are chosen as 0.7 and 0.1 for negative class and -0.6 and 1 for positive class. Fig. 3.2 shows the distribution of data points in the Crossplane dataset. AUC values and training time are shown in Table 3.2 for RBF kernel with the corresponding average ranks, for the performance comparison of the proposed RFLSTSVM-CIL with EFSVM, TWSVM, FTWSVM$_{lin}$, FTWSVM$_{exp}$, UTSVM, LSTSVM, FLSTSVM-CIL$_{lin}$ and FLSTSVM-CIL$_{exp}$ on the synthetic datasets.

Table 3.1: Imbalance ratio (IR) of synthetic imbalance datasets for all the samples and for the training data.

Dataset (Train size, Test size)	**Imbalance ratio** (All samples)	**Imbalance ratio** (Training samples)
04clover5z-600-5-60-BI (200×2, 400×2)	5	5.06
03subcl5-600-5-30-BI (200×2, 400×2)	5	4.71
03subcl5-600-5-50-BI (200×2, 400×2)	5	4.41
03subcl5-600-5-60-BI (200×2, 400×2)	5	5.06
Crossplane_400 (119×2, 281×2)	7	6
Crossplane_450 (134×2, 316×2)	8	7.93

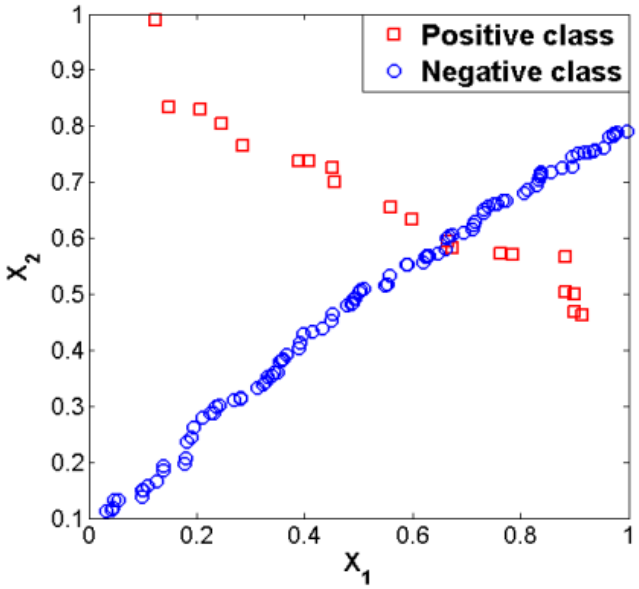


Figure 3.2: Plot showing Crossplane dataset containing 120 samples with imbalance ratio, $IR = 5$.

Table 3.2: Performance comparison of proposed RFLSTSVM-CIL on AUC and training time with EFSVM, TWSVM, FTWSVM$_{lin}$, FTWSVM$_{exp}$, UTSVM, LSTSVM, FLSTSVM-CIL$_{lin}$ and FLSTSVM-CIL$_{exp}$ using RBF kernel for classification on synthetic imbalance datasets. Average ranks are calculated on the basis of AUC. Bold values indicate best performance for the dataset.

Dataset (Train size, Test size)	**EFSVM** AUC (%) ± SD (c, μ, K) Time (s)	**TWSVM** AUC (%) ± SD (c, μ) Time (s)	**FTWSVM$_{lin}$** AUC (%) ± SD (c, μ) Time (s)	**FTWSVM$_{exp}$** AUC (%) ± SD (c, μ, β) Time (s)	**UTSVM** AUC (%) ± SD (c, μ, ϵ) Time (s)	**LSTSVM** AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM-CIL$_{lin}$** AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM-CIL$_{exp}$** AUC (%) ± SD (c, μ, β) Time (s)	**RFLSTSVM-CIL** AUC (%) ± SD (c, μ, c_0) Time (s)
04clover5z-600-5-60-BI (200×2, 400×2)	69.38 ± 9.52 $(10^{-5}, 2^5, 7)$ 1.1949	61.15 ± 8.13 $(10^{-3}, 2^5)$ 0.1408	63.36 ± 8.93 $(10^{-3}, 2^5)$ 0.1469	60.93 ± 6.36 $(10^0, 2^5, 0.1)$ 0.1269	64.33 ± 8.37 $(10^{-3}, 2^5, 0.3)$ 0.1532	72.28 ± 9.5 $(10^{-5}, 2^5)$ 0.1007	72.33 ± 5.52 $(10^{-5}, 2^5)$ 0.1115	67.35 ± 14.88 $(10^{-5}, 2^5, 0.5)$ 0.156	**75.54 ± 8.33** $(10^{-4}, 2^5, 0.5)$ 0.1122
03subcl5-600-5-30-BI (200×2, 400×2)	71.26 ± 6.1 $(10^2, 2^5, 3)$ 1.2436	65.83 ± 6.76 $(10^{-4}, 2^5)$ 0.1442	64.66 ± 6.63 $(10^{-4}, 2^5)$ 0.1566	60.25 ± 4.43 $(10^5, 2^5, 0.3)$ 0.1392	67.62 ± 7.79 $(10^0, 2^5, 0.3)$ 0.1736	66.72 ± 5.55 $(10^{-4}, 2^5)$ 0.1111	72.64 ± 5.55 $(10^{-4}, 2^5)$ 0.1297	**75.45 ± 10** $(10^{-5}, 2^5, 0.3)$ 0.1537	72.8 ± 2.86 $(10^{-2}, 2^5, 1.5)$ 0.1264
03subcl5-600-5-50-BI (200×2, 400×2)	58.65 ± 8.84 $(10^{-5}, 2^5, 3)$ 1.1921	51.86 ± 4.87 $(10^{-4}, 2^5)$ 0.1448	52.95 ± 4.28 $(10^{-3}, 2^5)$ 0.1505	49.57 ± 1.73 $(10^0, 2^5, 0.1)$ 0.1342	56.31 ± 10.4 $(10^{-2}, 2^5, 0.6)$ 0.1592	54.76 ± 8.58 $(10^{-4}, 2^5)$ 0.1102	55.7 ± 7.65 $(10^{-5}, 2^5)$ 0.1267	**60.35 ± 6.35** $(10^{-5}, 2^5, 0.1)$ 0.1255	58.1 ± 7.11 $(10^{-5}, 2^5, 0.5)$ 0.1283
03subcl5-600-5-60-BI (200×2, 400×2)	69.73 ± 5.75 $(10^0, 2^5, 3)$ 1.1877	66.18 ± 6.78 $(10^{-1}, 2^5)$ 0.1498	65.91 ± 6.18 $(10^{-1}, 2^5)$ 0.1504	62.98 ± 2.35 $(10^2, 2^5, 0.1)$ 0.131	66.84 ± 8.16 $(10^{-1}, 2^5, 0.3)$ 0.1624	71.04 ± 5.67 $(10^{-4}, 2^5)$ 0.1073	**72.75 ± 5.9** $(10^{-4}, 2^5)$ 0.1237	71.19 ± 7.63 $(10^{-5}, 2^5, 0.3)$ 0.1463	72.07 ± 5.8 $(10^{-4}, 2^5, 2)$ 0.1281
Crossplane400 (119×2, 281×2)	92.1 ± 9.14 $(10^0, 2^{-4}, 3)$ 0.6101	95.43 ± 6.97 $(10^{-5}, 2^{-1})$ 0.0743	93.97 ± 5.58 $(10^{-2}, 2^0)$ 0.0693	95.65 ± 3.24 $(10^{-2}, 2^0, 1)$ 0.0712	95.43 ± 6.97 $(10^{-5}, 2^{-1}, 0.1)$ 0.0719	93.99 ± 1.65 $(10^{-3}, 2^{-1})$ 0.0547	**98.58 ± 1.49** $(10^{-4}, 2^{-1})$ 0.0702	94.59 ± 2.12 $(10^{-3}, 2^{-1}, 0.7)$ 0.0545	97.18 ± 1.5 $(10^{-1}, 2^2, 2.5)$ 0.0552
Crossplane450 (134×2, 316×2)	96.23 ± 6.2 $(10^0, 2^{-5}, 3)$ 0.7515	97.32 ± 3.68 $(10^{-5}, 2^1)$ 0.0864	97.32 ± 3.68 $(10^{-5}, 2^1)$ 0.0839	97.32 ± 3.68 $(10^{-5}, 2^1, 0.1)$ 0.0852	97.32 ± 3.68 $(10^{-5}, 2^1, 0.1)$ 0.089	98.96 ± 1.12 $(10^{-5}, 2^{-3})$ 0.0686	99.05 ± 1.42 $(10^{-5}, 2^{-3})$ 0.0904	98.96 ± 1.12 $(10^{-5}, 2^{-3}, 0.1)$ 0.0702	**99.13 ± 0.88** $(10^{-4}, 2^{-3}, 1.5)$ 0.0791
Average AUC	76.23	72.96	73.03	71.12	74.64	76.29	78.51	77.98	**79.14**
Average rank	5.5	6.8333	7.4167	7.5833	5.3333	4.9167	2.3333	3.25	**1.8333**

3.1.5.2 Real world datasets

Numerical experiments are performed on several real world imbalanced datasets for binary classification. The class imbalance ratios of the various real world datasets are shown in Table 3.3. The performance of the proposed RFLSTSVM-CIL is compared with existing algorithms in terms of AUC values and training time in Table 3.4.

Table 3.3: Imbalance ratio (IR) of real world datasets for all the samples and for the training data.

Dataset (Train size, Test size)	**Imbalance ratio** (All samples)	**Imbalance ratio** (Training samples)	**Dataset** (Train size, Test size)	**Imbalance ratio** (All samples)	**Imbalance ratio** (Training samples)
Cmc (700×9, 773×9)	0.75	1.46	Yeast-0-5-6-7-9_vs_4 (250×8, 278×8)	9.35	8.26
Ecoli-0-1_vs_2-3-5 (120×7, 124×7)	9.17	9	Ecoli-0-1-4-6_vs_5 (150×6, 130×6)	13	9.71
Ecoli-0-1_vs_5 (120×6, 120×6)	11	16.14	Ecoli2 (150×7, 186×7)	8.6	6.89
Ecoli-0-1-4-7_vs_5-6 (150×6, 182×6)	12.28	10.54	Vowel (500×10, 488×10)	9.98	9.87
Ecoli-0-2-3-4_vs_5 (100×7, 102×7)	9.1	6.69	Ecoli3 (150×7, 186×7)	8.6	6.89
Ecoli-0-2-6-7_vs_3-5 (110×7, 114×7)	9.18	7.46	Abalone9-18 (350×7, 381×7)	16.4	18.44
Ecoli-0-3-4-6_vs_5 (100×7, 105×7)	9.25	8.09	Vehicle 1 (400×18, 446×18)	2.9	3.3
Ecoli-0-4-6_vs_5 (100×6, 103×6)	9.15	11.5	Vehicle2 (400×18, 446×18)	2.88	2.48
Ecoli-0-6-7_vs_3-5 (110×7, 112×7)	9.09	12.75	Pima-Indians (300×8, 468×8)	1.87	1.63
Ecoli-0-6-7_vs_5 (110×6, 110×6)	10	9	Yeast3 (500×8, 984×8)	8.1	7.2
Ecoli4 (150×7, 186×7)	15.8	17.75	Yeast1vs7 (200×8, 259×8)	14.3	15.67
Glass-0-4_vs_5 (50×9, 42×9)	9.22	6.14	Yeast2vs8 (250×8, 233×8)	23.15	19.83
Glass2 (100×9, 114×9)	11.59	13.29	Ecoli0137vs26 (180×7, 131×7)	4.76	4.63
Ripley (600×2, 650×2)	1	1.08	Australian-Credit (300×14, 390×14)	1.25	1.13
Yeast-0-2-5-6_vs_3-7-8-9 (500×8, 504×8)	9.14	11.5	Monk2 (300×7, 301×7)	1.92	2.06
Yeast-0-3-5-9_vs_7-8 (250×8, 256×8)	9.12	11.5			

Table 3.4: Performance comparison of proposed RFLSTSVM-CIL on average AUC and training time with EFSVM, TWSVM, FTWSVM$_{lin}$, FTWSVM$_{exp}$, UTSVM, LSTSVM, FLSTSVM-CIL$_{lin}$ and FLSTSVM-CIL$_{exp}$ using RBF kernel for classification on real world datasets. Average ranks are calculated on the basis of AUC.

Dataset (Train size, Test size)	**EFSVM** AUC (%) ± SD (c, μ, K) Time (s)	**TWSVM** AUC (%) ± SD (c, μ) Time (s)	**FTWSVM$_{lin}$** AUC (%) ± SD (c, μ) Time (s)	**FTWSVM$_{exp}$** AUC (%) ± SD (c, μ, β) Time (s)	**UTSVM** AUC (%) ± SD (c, μ, ϵ) Time (s)	**LSTSVM** AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM-CIL$_{lin}$** AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM-CIL$_{exp}$** AUC (%) ± SD (c, μ, β) Time (s)	**RFLSTSVM-CIL** AUC (%) ± SD (c, μ, c_0) Time (s)
Cmc (700×9, 773×9)	64.31 ± 29.12 $(10^4, 2^2, 9)$ 4.6819	69.13 ± 34.63 $(10^{-2}, 2^4)$ 0.4954	**70.68 ± 32.08** $(10^{-5}, 2^5)$ 0.5306	68.56 ± 32.92 $(10^{-5}, 2^4, 0.1)$ 0.5227	53 ± 4.05 $(10^{-1}, 2^{-1}, 0.5)$ 0.6908	59.02 ± 17.43 $(10^{-2}, 2^4)$ 0.3974	57.45 ± 14.58 $(10^{-2}, 2^4)$ 0.4448	59.33 ± 15.84 $(10^{-2}, 2^4, 0.3)$ 0.4398	60.66 ± 19.64 $(10^{-2}, 2^4, 0.5)$ 0.4521
Ecoli-0-1_vs_2-3-5 (120×7, 124×7)	77.42 ± 21.94 $(10^0, 2^5, 7)$ 0.1211	67.42 ± 21.77 $(10^{-5}, 2^5)$ 0.0165	67.42 ± 21.77 $(10^{-3}, 2^5)$ 0.0186	61.53 ± 21.48 $(10^{-4}, 2^5, 0.1)$ 0.019	72.42 ± 26.33 $(10^{-5}, 2^5, 0.1)$ 0.0195	69.55 ± 22.27 $(10^{-5}, 2^4)$ 0.0102	**84.32 ± 11.65** $(10^{-5}, 2^4)$ 0.011	75.76 ± 26.48 $(10^{-4}, 2^5, 0.3)$ 0.0107	80.32 ± 13.29 $(10^{-5}, 2^5, 0.5)$ 0.0113
Ecoli-0-1_vs_5 (120×6, 120×6)	78.71 ± 18.02 $(10^{-5}, 2^1, 3)$ 0.1116	76.25 ± 18.84 $(10^{-5}, 2^5)$ 0.0158	75.83 ± 19.18 $(10^{-5}, 2^4)$ 0.0163	75.83 ± 19.18 $(10^{-5}, 2^4, 0.1)$ 0.0159	72.88 ± 15.73 $(10^{-3}, 2^4, 0.5)$ 0.0177	**83.75 ± 23.22** $(10^{-5}, 2^5)$ 0.0092	81.21 ± 22.11 $(10^{-3}, 2^5)$ 0.0102	74.97 ± 16.96 $(10^{-5}, 2^4, 0.5)$ 0.0101	82.88 ± 23.74 $(10^{-4}, 2^5, 0.5)$ 0.0117
Ecoli-0-1-4-7_vs_5-6 (150×6, 182×6)	83.33 ± 20.41 $(10^0, 2^5, 5)$ 0.2588	76.67 ± 18.07 $(10^{-4}, 2^4)$ 0.0311	68.33 ± 20.75 $(10^{-2}, 2^5)$ 0.0381	76.34 ± 18.12 $(10^3, 2^4, 1)$ 0.0302	60 ± 22.36 $(10^{-2}, 2^3, 0.5)$ 0.036	**83.81 ± 19.86** $(10^{-3}, 2^5)$ 0.0208	75.76 ± 18.22 $(10^{-5}, 2^4)$ 0.0238	76.34 ± 18.12 $(10^{-5}, 2^4, 1)$ 0.0225	**83.81 ± 19.86** $(10^{-3}, 2^5, 0.5)$ 0.0234
Ecoli-0-2-3-4_vs_5 (100×7, 102×7)	**99.5 ± 1.12** $(10^0, 2^4, 3)$ 0.0826	85 ± 22.36 $(10^{-1}, 2^5)$ 0.0126	85 ± 22.36 $(10^{-1}, 2^5)$ 0.0135	75 ± 25 $(10^{-4}, 2^4, 0.3)$ 0.0121	74.21 ± 13.82 $(10^{-2}, 2^3, 0.5)$ 0.0138	**99.5 ± 1.12** $(10^{-5}, 2^5)$ 0.0067	98.42 ± 1.45 $(10^{-5}, 2^5)$ 0.0102	90 ± 22.36 $(10^{-5}, 2^4, 1)$ 0.0074	**99.5 ± 1.12** $(10^{-4}, 2^5, 2.5)$ 0.0078
Ecoli-0-2-6-7_vs_3-5 (110×7, 114×7)	63.33 ± 12.64 $(10^0, 2^5, 3)$ 0.1022	63.33 ± 12.64 $(10^{-3}, 2^5)$ 0.0167	65.83 ± 16.24 $(10^{-3}, 2^5)$ 0.0154	62.81 ± 12.07 $(10^{-5}, 2^5, 0.1)$ 0.0163	61.95 ± 13.43 $(10^{-3}, 2^3, 0.6)$ 0.0174	**66.13 ± 24.85** $(10^{-2}, 2^5)$ 0.0105	63.92 ± 17.45 $(10^{-3}, 2^5)$ 0.0096	59.07 ± 13.51 $(10^{-5}, 2^3, 1)$ 0.0111	64.38 ± 16.87 $(10^{-3}, 2^5, 1.5)$ 0.0118
Ecoli-0-3-4-6_vs_5 (100×7, 105×7)	**81.47 ± 20.68** $(10^0, 2^5, 3)$ 0.0876	76.47 ± 17.91 $(10^{-5}, 2^5)$ 0.0136	80.97 ± 20.22 $(10^{-5}, 2^5)$ 0.0142	79.47 ± 21.11 $(10^4, 2^5, 1)$ 0.0128	76.47 ± 17.91 $(10^{-5}, 2^5, 0.1)$ 0.0151	80 ± 20.73 $(10^{-3}, 2^5)$ 0.009	79.47 ± 21.11 $(10^0, 2^5)$ 0.0095	78.47 ± 22.18 $(10^{-5}, 2^5, 0.1)$ 0.0086	81 ± 21.59 $(10^{-4}, 2^5, 1.5)$ 0.0114
Ecoli-0-4-6_vs_5 (100×6, 103×6)	84.44 ± 14.25 $(10^{-1}, 2^4, 3)$ 0.0842	84.44 ± 15.04 $(10^{-4}, 2^5)$ 0.0125	84.44 ± 15.04 $(10^{-3}, 2^5)$ 0.0133	86.11 ± 19.04 $(10^{-5}, 2^4, 0.1)$ 0.013	73.45 ± 19.73 $(10^{-2}, 2^4, 0.6)$ 0.0141	88.36 ± 14.38 $(10^{-5}, 2^5)$ 0.0072	88.92 ± 14.85 $(10^{-5}, 2^5)$ 0.0098	**94.47 ± 6.53** $(10^{-5}, 2^4, 0.7)$ 0.0093	88.92 ± 14.85 $(10^{-5}, 2^5, 0.5)$ 0.0115
Ecoli-0-6-7_vs_3-5 (110×7, 112×7)	77.78 ± 15.49 $(10^{-5}, 2^3, 3)$ 0.0996	73.28 ± 18.42 $(10^{-3}, 2^4)$ 0.0141	80.78 ± 13.01 $(10^{-3}, 2^4)$ 0.0164	78.78 ± 12.26 $(10^{-4}, 2^4, 0.3)$ 0.0137	74.17 ± 3.12 $(10^{-1}, 2^4, 0.6)$ 0.016	83.33 ± 13.48 $(10^{-5}, 2^5)$ 0.0094	**83.83 ± 13.44** $(10^{-5}, 2^5)$ 0.0089	80.83 ± 13.74 $(10^{-3}, 2^5, 0.1)$ 0.011	82.83 ± 13.9 $(10^{-4}, 2^5, 0.5)$ 0.0104
Ecoli-0-6-7_vs_5 (110×6, 110×6)	73 ± 16.9 $(10^{-5}, 2^3, 5)$ 0.0962	59 ± 22.95 $(10^{-1}, 2^5)$ 0.0141	59 ± 22.95 $(10^{-1}, 2^5)$ 0.0148	69 ± 19.89 $(10^2, 2^5, 0.3)$ 0.015	73.5 ± 24.47 $(10^{-2}, 2^4, 0.6)$ 0.0154	**86 ± 12.94** $(10^{-3}, 2^5)$ 0.0081	82 ± 21.17 $(10^{-4}, 2^5)$ 0.0085	76.02 ± 23.65 $(10^{-5}, 2^4, 0.1)$ 0.0109	81 ± 20.2 $(10^{-3}, 2^5, 0.5)$ 0.0088

Table 3.4 (contd.)

Dataset (Train size, Test size)	**EFSVM** AUC (%) ± SD (c, μ, K) Time (s)	**TWSVM** AUC (%) ± SD (c, μ) Time (s)	**FTWSVM$_{lin}$** AUC (%) ± SD (c, μ) Time (s)	**FTWSVM$_{exp}$** AUC (%) ± SD (c, μ, β) Time (s)	**UTSVM** AUC (%) ± SD (c, μ, ϵ) Time (s)	**LSTSVM** AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM$_{lin}$** AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM$_{exp}$** AUC (%) ± SD (c, μ, β) Time (s)	**RFLSTSVM-CIL** AUC (%) ± SD (c, μ, c_0) Time (s)
Ecoli4 (150×7, 186×7)	93.87 ± 8.62 $(10^1, 2^{-2}, 3)$ 0.2707	96.34 ± 5.89 $(10^0, 2^2)$ 0.0316	96.63 ± 5.97 $(10^0, 2^2)$ 0.0397	96.34 ± 5.89 $(10^0, 2^2, 0.7)$ 0.0347	93.87 ± 8.62 $(10^{-2}, 2^{-1}, 0.6)$ 0.037	95.76 ± 7.11 $(10^{-3}, 2^{-1})$ 0.0226	**97.12 ± 1.03** $(10^{-2}, 2^4)$ 0.0235	95.76 ± 7.11 $(10^{-3}, 2^{-1}, 0.1)$ 0.0242	97.11 ± 1.07 $(10^{-5}, 2^1, 0.5)$ 0.0236
Glass-0-4_vs_5 (50×9, 42×9)	68.57 ± 25.56 $(10^1, 2^1, 3)$ 0.0163	60 ± 22.36 $(10^{-1}, 2^4)$ 0.0066	68.57 ± 25.56 $(10^{-1}, 2^4)$ 0.0071	60 ± 22.36 $(10^{-1}, 2^4, 0.1)$ 0.007	68.57 ± 25.56 $(10^0, 2^4, 0.6)$ 0.0058	**70 ± 27.39** $(10^2, 2^0)$ 0.0013	**70 ± 27.39** $(10^{-5}, 2^3)$ 0.0026	**70 ± 27.39** $(10^{-3}, 2^3, 0.5)$ 0.0022	**70 ± 27.39** $(10^0, 2^3, 0.5)$ 0.0016
Glass2 (100×9, 114×9)	53.33 ± 22.29 $(10^5, 2^2, 3)$ 0.1039	51.15 ± 10.31 $(10^{-3}, 2^3)$ 0.0145	47.79 ± 10.1 $(10^{-3}, 2^3)$ 0.0153	46.88 ± 9.33 $(10^{-3}, 2^3, 0.3)$ 0.0151	52.19 ± 15.25 $(10^3, 2^3, 0.6)$ 0.0166	62.5 ± 17.4 $(10^{-3}, 2^2)$ 0.0085	56.34 ± 18.37 $(10^{-3}, 2^3)$ 0.0128	**66.81 ± 16.59** $(10^{-4}, 2^{-1}, 1)$ 0.0122	58.16 ± 17.63 $(10^{-2}, 2^3, 1.5)$ 0.0093
Ripley (600×2, 650×2)	91.27 ± 2.71 $(10^0, 2^{-2}, 7)$ 3.2518	**91.54 ± 1.95** $(10^{-1}, 2^{-1})$ 0.3452	91.48 ± 2.14 $(10^{-1}, 2^{-1})$ 0.3485	**91.54 ± 2.58** $(10^{-1}, 2^{-1}, 1)$ 0.3531	90.23 ± 1.91 $(10^0, 2^{-1}, 0.1)$ 0.4037	90.64 ± 2.98 $(10^0, 2^1)$ 0.2766	90.8 ± 2.4 $(10^0, 2^{-1})$ 0.3092	91.43 ± 2.13 $(10^0, 2^{-1}, 0.1)$ 0.3033	91.03 ± 2.3 $(10^0, 2^0, 1.5)$ 0.2999
Yeast-0-2-5-6_vs_3-7-8-9 (500×8, 504×8)	64.99 ± 5.44 $(10^2, 2^0, 7)$ 2.0021	73.23 ± 8.09 $(10^{-5}, 2^3)$ 0.2059	74.43 ± 5.1 $(10^{-2}, 2^2)$ 0.2146	72.39 ± 7.12 $(10^{-5}, 2^2, 0.5)$ 0.2155	74.42 ± 5.92 $(10^1, 2^1, 0.6)$ 0.2759	**79.1 ± 7.74** $(10^{-1}, 2^2)$ 0.1678	73.06 ± 6.8 $(10^0, 2^3)$ 0.1802	**79.1 ± 7.74** $(10^{-1}, 2^2, 0.1)$ 0.179	75.29 ± 8.19 $(10^0, 2^4, 1.5)$ 0.1798
Yeast-0-3-5-9_vs_7-8 (250×8, 256×8)	55.81 ± 5.95 $(10^3, 2^{-2}, 7)$ 0.5044	58.45 ± 6.08 $(10^{-2}, 2^0)$ 0.0552	63.33 ± 3.58 $(10^{-2}, 2^0)$ 0.0577	60.74 ± 8.25 $(10^{-2}, 2^0, 0.1)$ 0.0586	56.19 ± 10.96 $(10^2, 2^{-2}, 0.6)$ 0.0709	66.92 ± 6.84 $(10^{-1}, 2^1)$ 0.0459	65.14 ± 6.26 $(10^{-1}, 2^1)$ 0.0464	66.92 ± 6.84 $(10^{-1}, 2^1, 0.1)$ 0.045	**73.15 ± 2.97** $(10^0, 2^3, 2.5)$ 0.0463
Yeast-0-5-6-7-9_vs_4 (250×8, 278×8)	66.73 ± 6.6 $(10^3, 2^1, 3)$ 0.5954	65.35 ± 6.83 $(10^{-1}, 2^3)$ 0.066	68.27 ± 10.72 $(10^{-1}, 2^3)$ 0.0692	65.35 ± 6.83 $(10^{-1}, 2^3, 0.1)$ 0.0681	50 ± 0 $(10^1, 2^1, 0.1)$ 0.0797	**82.27 ± 12.89** $(10^{-3}, 2^{-1})$ 0.0497	72.92 ± 16.43 $(10^0, 2^0)$ 0.0558	81.43 ± 14.11 $(10^{-3}, 2^{-1}, 1)$ 0.0523	76.35 ± 15.84 $(10^{-2}, 2^{-1}, 2)$ 0.0529
Ecoli-0-1-4-6_vs_5 (150×6, 130×6)	100 ± 0 $(10^{-5}, 2^3, 3)$ 0.131	95 ± 11.18 $(10^{-3}, 2^5)$ 0.0182	95 ± 11.18 $(10^{-2}, 2^5)$ 0.0192	95 ± 11.18 $(10^3, 2^5, 0.3)$ 0.0252	**100 ± 0** $(10^{-3}, 2^5, 0.5)$ 0.0205	**100 ± 0** $(10^{-2}, 2^5)$ 0.0106	**100 ± 0** $(10^{-5}, 2^4)$ 0.012	**100 ± 0** $(10^4, 2^4, 0.5)$ 0.012	**100 ± 0** $(10^{-2}, 2^5, 2.5)$ 0.0135
Ecoli2 (150×7, 186×7)	84.61 ± 22.4 $(10^0, 2^{-2}, 11)$ 0.2656	82.37 ± 9.02 $(10^{-5}, 2^4)$ 0.0319	85.12 ± 8.07 $(10^{-5}, 2^4)$ 0.035	82.37 ± 9.02 $(10^{-5}, 2^4, 0.1)$ 0.0323	82.37 ± 9.02 $(10^{-5}, 2^4, 0.1)$ 0.0341	**86.42 ± 9.5** $(10^{-1}, 2^5)$ 0.0281	83.7 ± 4.62 $(10^{-2}, 2^2)$ 0.0236	**86.42 ± 9.5** $(10^{-1}, 2^5, 0.1)$ 0.0233	85.14 ± 7.64 $(10^{-2}, 2^1, 2.5)$ 0.0244
Vowel (500×10, 488×10)	85.1 ± 9.92 $(10^1, 2^3, 11)$ 1.8297	95.48 ± 4.73 $(10^{-1}, 2^5)$ 0.2214	93.93 ± 8.02 $(10^{-1}, 2^5)$ 0.2191	95.48 ± 4.73 $(10^{-1}, 2^5, 0.1)$ 0.2306	89.32 ± 12.11 $(10^0, 2^4, 0.3)$ 0.2602	95.26 ± 7.21 $(10^{-1}, 2^5)$ 0.1569	95.6 ± 5.52 $(10^{-2}, 2^4)$ 0.169	93.12 ± 7.87 $(10^0, 2^5, 0.7)$ 0.1709	**96.17 ± 6.34** $(10^{-1}, 2^5, 0.5)$ 0.171
Ecoli3 (150×7, 186×7)	84.61 ± 22.4 $(10^0, 2^{-2}, 11)$ 0.271	82.37 ± 9.02 $(10^{-5}, 2^4)$ 0.0308	85.12 ± 8.07 $(10^{-5}, 2^4)$ 0.0368	82.37 ± 9.02 $(10^{-5}, 2^4, 0.1)$ 0.0333	82.37 ± 9.02 $(10^{-5}, 2^4, 0.1)$ 0.034	**86.42 ± 9.5** $(10^{-1}, 2^5)$ 0.0218	83.7 ± 4.62 $(10^{-2}, 2^2)$ 0.024	**86.42 ± 9.5** $(10^{-1}, 2^5, 0.1)$ 0.0236	85.14 ± 7.64 $(10^{-2}, 2^1, 2.5)$ 0.0239

Table 3.4 (contd.)

Dataset (Train size, Test size)	**EFSVM** AUC (%) ± SD (c, μ, K) Time (s)	**TWSVM** AUC (%) ± SD (c, μ) Time (s)	**FTWSVM**$_{lin}$ AUC (%) ± SD (c, μ) Time (s)	**FTWSVM**$_{exp}$ AUC (%) ± SD (c, μ, β) Time (s)	**UTSVM** AUC (%) ± SD (c, μ, ϵ) Time (s)	**LSTSVM** AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM-CIL**$_{lin}$ AUC (%) ± SD (c, μ) Time (s)	**FLSTSVM-CIL**$_{exp}$ AUC (%) ± SD (c, μ, β) Time (s)	**RFLSTSVM-CIL** AUC (%) ± SD (c, μ, c_0) Time (s)
Abalone9-18 (350×7, 381×7)	67.62 ± 10.73 $(10^5, 2^2, 7)$ 1.1422	71.19 ± 18.71 $(10^{-2}, 2^0)$ 0.1277	66.87 ± 11.88 $(10^{-1}, 2^{-1})$ 0.1346	66.69 ± 10.65 $(10^{-1}, 2^0, 0.7)$ 0.1384	66.41 ± 10.91 $(10^{-1}, 2^0, 0.5)$ 0.1489	72.52 ± 17.89 $(10^{-1}, 2^1)$ 0.0931	76.32 ± 17.83 $(10^{-1}, 2^1)$ 0.1	73.95 ± 18.48 $(10^{-1}, 2^1, 0.3)$ 0.0997	**79.53 ± 19.71** $(10^0, 2^1, 1)$ 0.0996
Vehicle 1 (400×18, 446×18)	69.96 ± 3.23 $(10^1, 2^5, 7)$ 1.5729	66.6 ± 5.59 $(10^{-5}, 2^5)$ 0.1684	67.66 ± 4.1 $(10^{-2}, 2^5)$ 0.1743	64.55 ± 6.25 $(10^4, 2^5, 0.1)$ 0.1814	67.87 ± 4.59 $(10^{-2}, 2^5, 0.3)$ 0.1912	**70.52 ± 6.59** $(10^{-3}, 2^5)$ 0.138	67.43 ± 5.56 $(10^{-5}, 2^4)$ 0.1511	66.18 ± 4.9 $(10^{-4}, 2^5, 0.1)$ 0.1467	69.86 ± 7.57 $(10^{-3}, 2^5, 2.5)$ 0.1467
Vehicle2 (400×18, 446×18)	92.96 ± 3.46 $(10^1, 2^5, 11)$ 1.5622	85.38 ± 6.69 $(10^{-2}, 2^5)$ 0.1721	86.56 ± 7.03 $(10^{-2}, 2^5)$ 0.1795	83.45 ± 6.98 $(10^4, 2^5, 0.1)$ 0.1946	91.2 ± 3.27 $(10^{-1}, 2^5, 0.5)$ 0.1995	93.25 ± 4.28 $(10^{-1}, 2^5)$ 0.1382	92.59 ± 4.5 $(10^{-1}, 2^5)$ 0.1442	86.31 ± 1.98 $(10^{-3}, 2^5, 0.1)$ 0.1469	**93.39 ± 5.45** $(10^{-2}, 2^5, 0.5)$ 0.1469
Pima-Indians (300×8, 468×8)	71.47 ± 2.39 $(10^{-1}, 2^5, 11)$ 1.6708	66.32 ± 3.08 $(10^{-5}, 2^5)$ 0.1759	68.04 ± 4.15 $(10^{-5}, 2^5)$ 0.1825	60.69 ± 4.35 $(10^{-1}, 2^5, 0.1)$ 0.1851	50 ± 0 $(10^0, 2^0, 0.3)$ 0.212	71.5 ± 1.82 $(10^{-4}, 2^5)$ 0.1461	72.4 ± 2.89 $(10^{-4}, 2^5)$ 0.1586	70.92 ± 1.15 $(10^{-3}, 2^5, 0.3)$ 0.1567	**72.95 ± 4.14** $(10^{-3}, 2^5, 2)$ 0.1554
Yeast3 (500×8, 984×8)	79.94 ± 4.81 $(10^1, 2^{-3}, 5)$ 7.5527	86.51 ± 4.98 $(10^{-2}, 2^2)$ 0.9194	85.68 ± 4.67 $(10^{-2}, 2^2)$ 0.9642	86.45 ± 5.07 $(10^{-2}, 2^2, 1)$ 0.9509	88.25 ± 2.8 $(10^1, 2^{-1}, 0.5)$ 1.1206	90.91 ± 4.11 $(10^{-2}, 2^0)$ 0.6599	**91.54 ± 4.56** $(10^{-1}, 2^0)$ 0.72	90.62 ± 3.94 $(10^{-2}, 2^0, 1)$ 0.7324	90.26 ± 3.81 $(10^{-1}, 2^3, 1.5)$ 0.7326
Yeast1vs7 (200×8, 259×8)	**70.21 ± 22.98** $(10^2, 2^{-2}, 3)$ 0.5025	68.92 ± 18.56 $(10^{-3}, 2^2)$ 0.0569	63.89 ± 17.07 $(10^{-3}, 2^1)$ 0.0612	68.92 ± 18.56 $(10^{-5}, 2^2, 0.3)$ 0.0583	67.46 ± 17.94 $(10^1, 2^{-1}, 0.5)$ 0.0683	59.54 ± 15.38 $(10^{-3}, 2^3)$ 0.0425	66.08 ± 6.35 $(10^{-2}, 2^2)$ 0.0456	58.7 ± 14.91 $(10^{-3}, 2^3, 0.3)$ 0.0452	62.76 ± 10.89 $(10^{-2}, 2^2, 0.5)$ 0.0456
Yeast2vs8 (250×8, 233×8)	58.12 ± 11.98 $(10^2, 2^{-2}, 7)$ 0.4211	**61.67 ± 16.24** $(10^{-5}, 2^2)$ 0.0468	**61.67 ± 16.24** $(10^{-5}, 2^2)$ 0.05	**61.67 ± 16.24** $(10^{-5}, 2^2, 0.3)$ 0.0546	**61.67 ± 16.24** $(10^0, 2^{-1}, 0.3)$ 0.0601	56.09 ± 14.83 $(10^{-5}, 2^{-3})$ 0.0347	58.73 ± 14.92 $(10^{-5}, 2^{-3})$ 0.0368	56.74 ± 14.94 $(10^{-5}, 2^{-3}, 1)$ 0.0373	58.72 ± 14.26 $(10^{-5}, 2^{-3}, 0.5)$ 0.0377
Ecoli0137vs26 (180×7, 131×7)	**98.35 ± 2.71** $(10^{-5}, 2^{-3}, 3)$ 0.1356	92.09 ± 10.82 $(10^{-5}, 2^4)$ 0.0168	95 ± 11.18 $(10^0, 2^0)$ 0.0185	95 ± 11.18 $(10^0, 2^0, 1)$ 0.018	96.33 ± 4.65 $(10^2, 2^4, 0.5)$ 0.0208	94.83 ± 4.29 $(10^{-1}, 2^0)$ 0.0138	91.7 ± 3.6 $(10^{-1}, 2^4)$ 0.0118	94.83 ± 4.29 $(10^{-1}, 2^0, 0.1)$ 0.0117	96.25 ± 8.39 $(10^0, 2^0, 1.5)$ 0.0122
Australian-Credit (300×14, 390×14)	86.39 ± 6.07 $(10^3, 2^4, 3)$ 1.1943	86.44 ± 5.21 $(10^{-2}, 2^3)$ 0.1245	86.64 ± 4.94 $(10^{-1}, 2^3)$ 0.1281	86.53 ± 2.58 $(10^{-5}, 2^4, 0.1)$ 0.127	85.82 ± 5.1 $(10^{-2}, 2^3, 0.1)$ 0.1383	85.16 ± 4.57 $(10^{-1}, 2^4)$ 0.1017	86.77 ± 6.01 $(10^0, 2^5)$ 0.1096	86.08 ± 5.62 $(10^0, 2^5, 0.7)$ 0.1095	**86.85 ± 5.92** $(10^0, 2^5, 1)$ 0.1092
Monk2 (300×7, 301×7)	50.26 ± 1.69 $(10^3, 2^2, 3)$ 0.6979	49.51 ± 1.3 $(10^{-1}, 2^2)$ 0.0769	51.9 ± 4.26 $(10^0, 2^2)$ 0.0831	51.03 ± 3.02 $(10^5, 2^2, 0.1)$ 0.087	45.33 ± 6.26 $(10^1, 2^3, 0.3)$ 0.0913	51.67 ± 2.75 $(10^{-1}, 2^3)$ 0.0592	55.86 ± 6.32 $(10^{-1}, 2^3)$ 0.0656	47.99 ± 6.49 $(10^{-3}, 2^3, 0.1)$ 0.0631	**57.4 ± 6.05** $(10^{-1}, 2^3, 0.5)$ 0.0656
Average AUC	76.69	74.74	75.54	74.42	72.64	79.38	78.81	77.9	**80.03**
Average rank	5.1129	6.1613	5.2258	6.2742	6.9677	3.6613	3.9677	4.8871	**2.7419**

One can notice from Table 3.4 that our proposed approach is having the least rank with less training time since our approach solves a pair of system of linear equations. To check the statistical significance of our proposed RFLSTSVM-CIL, we use Friedman test with the corresponding post-hoc test [172] for the 9 algorithms using 31 binary class datasets. Here, we assume that all the methods are equivalent under null hypothesis. The Friedman statistic is computed for the AUC values from Table 3.4 using Eq. (2.55). The χ^2 in this case is 61.4887. Then, the F_F value is calculated as 9.8903. Here, the F-distribution has $\big(9-1,(9-1)(31-1)\big) = (8, 240)$ degrees of freedom. Thus, for the significance level at $\alpha = 0.05$, the critical value for $F(8, 240)$ is 1.9771. Since $F_F = 6.8377 > 2.1595$, we reject the null hypothesis.

Now, to check the pairwise difference between the proposed and existing algorithms, we use the Nemenyi posthoc test. The critical difference is calculated using the formula in Eq. 2.56. For significant pairwise difference between the methods at significance level of $\alpha = 0.10$, the average ranks of the methods shown in Table 3.4 should differ by atleast $2.855\sqrt{\frac{9(9+1)}{6\times 31}} = 1.986$. The pairwise difference between the methods is shown in Table 3.5. The proposed RUTSVM-CIL is significantly better than most of the existing algorithms.

Table 3.5: Pairwise significant difference of proposed RFLSTSVM-CIL with existing algorithms.

Significance	**EFSVM**	**TWSVM**	**FTWSVM$_{lin}$**	**FTWSVM$_{exp}$**	**UTSVM**	**LSTSVM**	**FLSTSVM-CIL$_{lin}$**	**FLSTSVM-CIL$_{exp}$**
Proposed RFLSTSVM-CIL	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	No	**Yes**

3.1.6 Discussion

The proposed RFLSTSVM-CIL uses the 2-norm of the slack variables with the fuzzy membership values as shown in Eqs. (3.2) and (3.3). This makes the optimization problem strongly convex and gives globally optimal solution. For dealing with varying imbalance conditions, the novel fuzzy membership gives different ranges to the fuzzy membership values by using the information about the imbalance ratio (IR) of the data. The imbalance ratios of the synthetic and real datasets are shown

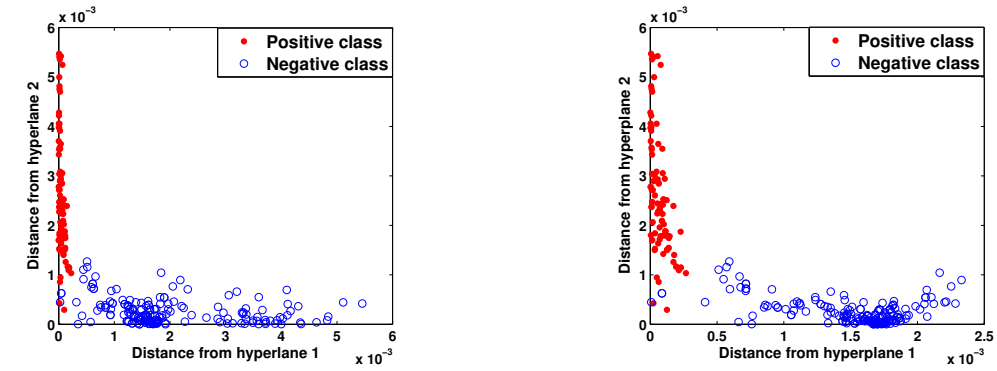
in Table 3.1 and Table 3.3 respectively. By incorporating the imbalance ratio (IR) in fuzzy function proper range is set for the fuzzy membership values on different datasets. Moreover, the information about the proximity of the data points to two classes is used in the proposed function which leads to better fuzzy membership for class imbalance data.

For the experiments on synthetic datasets, one can observe in Table 3.2 that the proposed RFLSTSVM-CIL is not performing better for all the synthetic datasets. However, RFLSTSVM-CIL is having the least ranks in most of the datasets, which justifies its robustness for different sets of data. For noisy data, our proposed method is having ranks as 1, 2, 3 and 2 out of 9 methods for 04clover5z-600-5-60-BI, 03subcl5-600-5-30-BI, 03subcl5-600-5-50-BI and 03subcl5-600-5-60-BI datasets respectively. Also, the training time of our proposed approach is lesser as compared to the existing algorithms in Table 3.2.

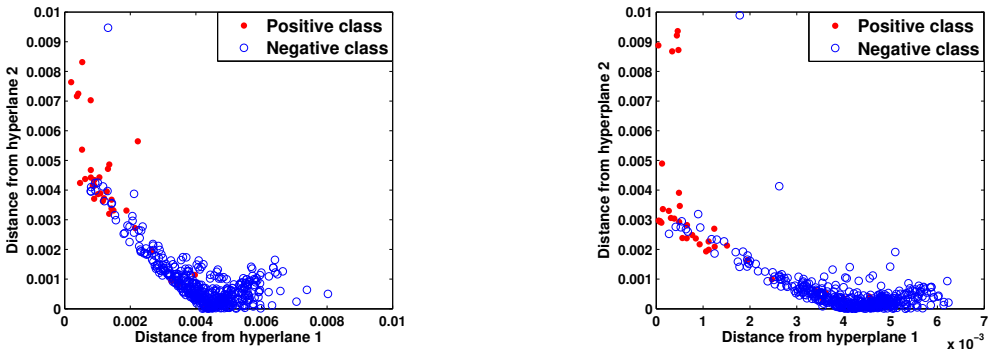
In real world datasets, our proposed RFLSTSVM-CIL is having the least rank with less training time in Table 3.4. It is observable that the proposed algorithm with the existing fuzzy functions i.e., FLSTSVM-CIL$_{lin}$ and FLSTSVM-CIL$_{exp}$ also perform better in comparison to the traditional approaches. The average ranks of FLSTSVM-CIL$_{lin}$ and FLSTSVM-CIL$_{exp}$ are lesser in comparison to EFSVM, TWSVM, FTWSVM$_{lin}$, FTWSVM$_{exp}$ and UTSVM in Table 3.2 and Table 3.4. In comparison to the existing algorithms, our proposed RFLSTSVM-CIL takes more computation time in comparison to LSTSVM. This is due to the additional computation for calculating the fuzzy membership values in the proposed approach.

The performance of the proposed method is compared with LSTSVM for showing the effect of proposed fuzzy membership in Fig. 3.3 for Monk2, Yeast-0-2-5-6-vs_3-7-8-9, Abalone9-18 and Vowel datasets. The figures show the distance of the data points with the two hyperplanes. It is observable from Figs. 3.3 that the data points of the positive class are nearer to the positive class hyperplane and away from the negative class hyperplane. This justifies the fact the proposed fuzzy membership function is efficient in calculating the proper fuzzy membership value for imbalance datasets.

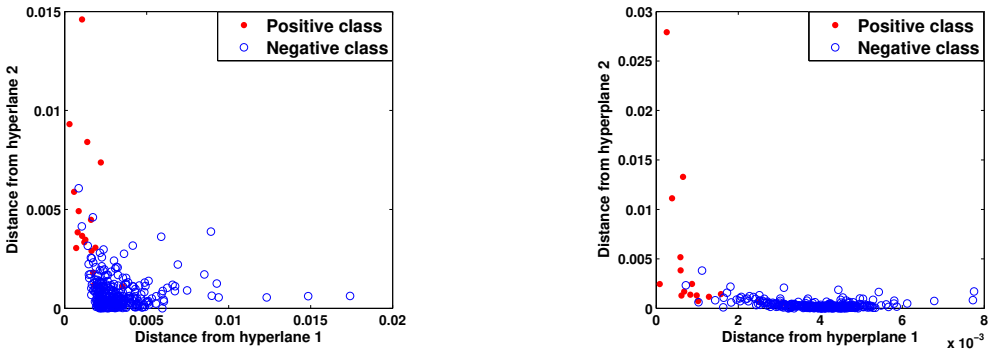
The insensitivity analysis of the proposed RFLSTSVM-CIL to the parameters c



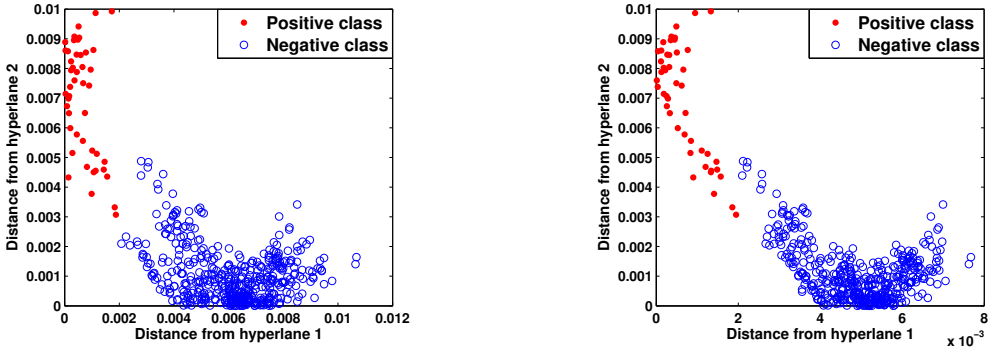
(a) Monk2



(b) Yeast-0-2-5-6-vs_3-7-8-9

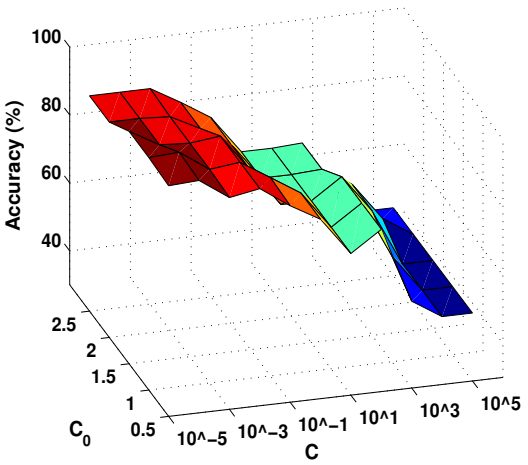


(c) Abalone9-18

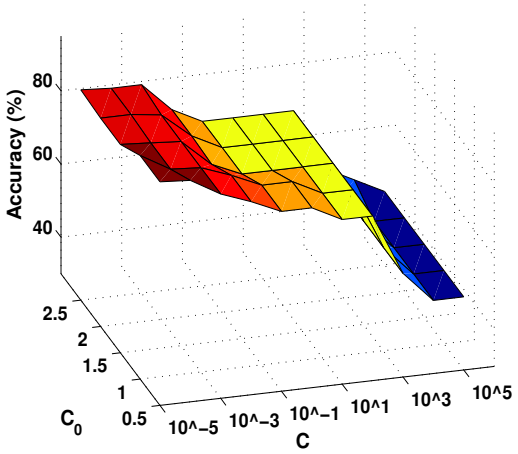


(d) Vowel

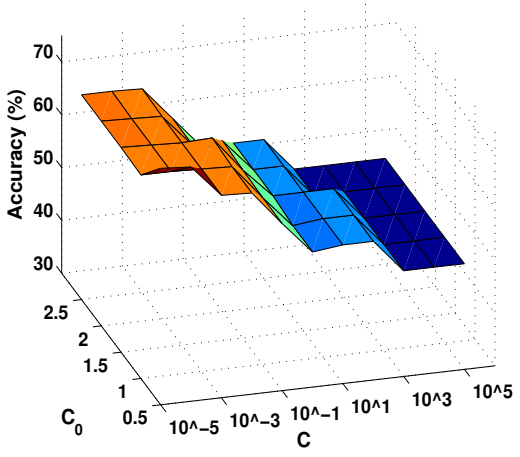
Figure 3.3: Distance of data points from the hyperplanes of proposed RFLSTSVM-CIL (left) and LSTSVM (right) for classification using RBF kernel.



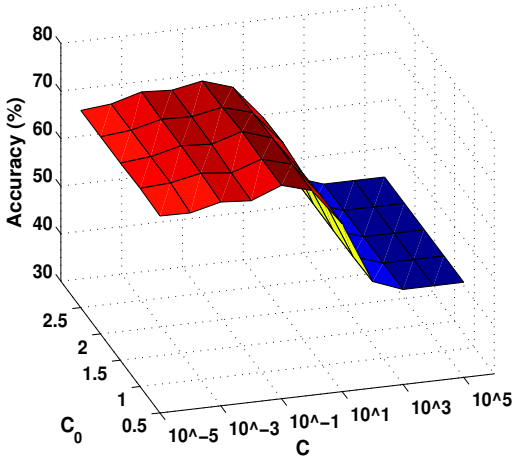
(a) Ecoli-0-3-4-6_vs_5



(b) Ecoli-0-4-6_vs_5



(c) Yeast2vs8



(d) Monk2

Figure 3.4: Insensitivity performance of the proposed RFLSTSVM-CIL for classification to the user specified parameters (c, c_0) using RBF kernel.

and c_0 is shown in Fig. 3.4 for Ecoli-0-3-4-6_vs_5, Ecoli-0-4-6_vs_5, Yeast2vs8 and Monk2 datasets. It can be observed that RFLSTSVM-CIL performs better on lesser values of c as well as c_0.

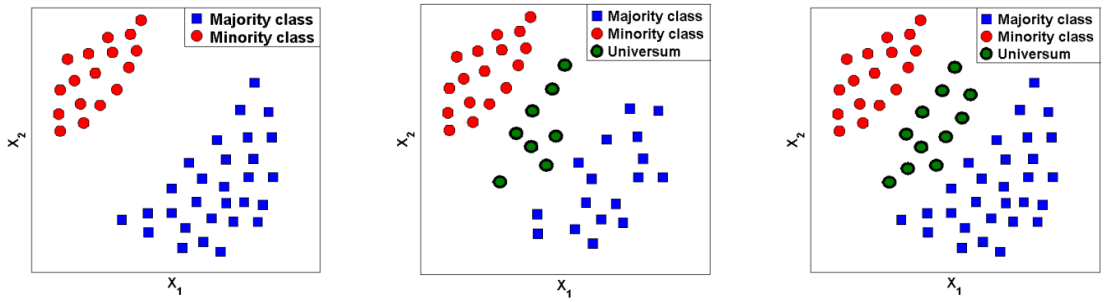
In the following section, we present a different approach for solving class imbalance. We propose a novel twin SVM algorithm by utilizing universum data and reduced kernel for removing the imbalance between the classes. This leads to an efficient as well as better model for classifying class imbalanced data.

3.2 A reduced universum twin support vector machine for class imbalance learning (RUTSVM-CIL)

The formulation of UTSVM [27] involves the solution of two small QPPs as compared to USVM [10] 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 [10,20,27]. 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 [179–181] as shown in Fig. 3.5(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 [20, 27].



(a) Imbalanced data (b) Case 2 (c) Case 3

Figure 3.5: 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.

3.2.1 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. 3.5(c), we treat universum data points as belonging to minority (positive) 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 $(1 - \epsilon)$ 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 $(r/2)$ 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.

3.2.2 Linear RUTSVM-CIL

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

$$\begin{aligned} \min_{w_1, b_1, \xi, \psi} \ & \frac{1}{2}\|X_1 w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi + c_u e_g^T \psi \\ s.t. \quad & -(X_2^* w_1 + e_1 b_1) + \xi \geq e_1, \\ & (U^* w_1 + e_g b_1) + \psi \geq (-1+\epsilon) e_g, \\ & \xi \geq 0, \quad \psi \geq 0, \end{aligned} \tag{3.16}$$

$$\begin{aligned} \min_{w_2, b_2, \eta, \psi^*} \ & \frac{1}{2}\|X_2 w_2 + e_2 b_2\|^2 + c_2 e_1^T \eta + c_u e_d^T \psi^* \\ s.t. \quad & (X_1 w_2 + e_1 b_2) + \eta \geq e_1, \\ & (U w_2 + e_d b_2) + \psi^* \geq (1-\epsilon) e_d, \\ & \eta \geq 0, \quad \psi^* \geq 0, \end{aligned} \tag{3.17}$$

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 (3.16) corresponds to the optimization problem for the positive class, and QPP (3.17) corresponds to the negative class hyperplane. One can observe that the constraint in (3.16) is to keep the universum within a distance of $(1-\epsilon)$ from the positive class hyperplane, and in QPP (3.17) the constraint is to keep the universum

away from the negative class hyperplane by a distance of $(1-\epsilon)$. Therefore, in QPP (3.16), the universum provides the prior information about the data, and in QPP (3.17) the prior information is provided in a manner that removes the biasing of the classifier.

The Wolfe duals of the primal problems (3.16) and (3.17) are obtained by applying the K.K.T. conditions as

$$\begin{aligned} &\max_{\alpha_1,\,\mu_1}\ e_1^T\alpha_1 - \frac{1}{2}(\alpha_1^T T^* - \mu_1^T O^*)(S^T S)^{-1}(T^{*T}\alpha_1 - O^{*T}\mu_1) + (\epsilon-1)e_g^T\mu_1 \\ &s.t.\ \ 0 \le \alpha_1 \le c_1, \quad 0 \le \mu_1 \le c_u, \end{aligned} \tag{3.18}$$

$$\begin{aligned} &\max_{\alpha_2,\,\mu_2}\ e_1^T\alpha_2 - \frac{1}{2}(\alpha_2^T S + \mu_2^T O)(T^T T)^{-1}(S^T\alpha_2 + O^T\mu_2) + (1-\epsilon)e_d^T\mu_2 \\ &s.t.\ \ 0 \le \alpha_2 \le c_2, \quad 0 \le \mu_2 \le c_u, \end{aligned} \tag{3.19}$$

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_i, b_i, $i = 1, 2$ from the following equations (3.20) and (3.21),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(S^T S + \sigma I)^{-1}(T^{*T}\alpha_1 - O^{*T}\mu_1), \tag{3.20}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (T^T T + \sigma I)^{-1}(S^T\alpha_2 + O^T\mu_2), \tag{3.21}$$

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 using Eq. (2.11).

3.2.3 Non-linear RUTSVM-CIL

The optimization problems of non-linear RUTSVM-CIL are expressed as follows:

$$\begin{aligned}\min_{w_1, b_1, \xi, \psi} \ & \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi + c_u e_g^T \psi \\ s.t. \quad & -(K(X_2^*, D^T)w_1 + e_1 b_1) + \xi \geq e_1, \\ & (K(U^*, D^T)w_1 + e_g b_1) + \psi \geq (-1+\epsilon)e_g, \\ & \xi \geq 0, \quad \psi \geq 0, \end{aligned} \tag{3.22}$$

$$\begin{aligned}\min_{w_2, b_2, \eta, \psi^*} \ & \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + c_2 e_1^T \eta + c_u e_d^T \psi^* \\ s.t. \quad & (K(X_1, D^T)w_2 + e_1 b_2) + \eta \geq e_1, \\ & (K(U, D^T)w_2 + e_d b_2) + \psi^* \geq (1-\epsilon)e_d, \\ & \eta \geq 0, \quad \psi^* \geq 0, \end{aligned} \tag{3.23}$$

where the slack variables are ξ, ψ, η, ψ^*; the penalty parameters are represented by c_1, c_2 and c_u; $D = [X_1^T \ X_2^{*T}]^T$, e_1, e_2, e_g and e_d are vectors of ones of appropriate dimension, and $K(x^T, D^T) = (k(x, x_1), k(x, x_1), \ldots, k(x, x_{2r}))$ is a row vector of the reduced kernel matrix in R^{2r} space, where r is the number of data points of 'class 1'.

The Wolfe duals of primal problems (3.22) and (3.23) are obtained using the K.K.T. conditions as

$$\begin{aligned}\max_{\alpha_1, \mu_1} \ & e_1^T \alpha_1 - \frac{1}{2}(\alpha_1^T F^* - \mu_1^T P^*)(E^T E)^{-1}(F^{*T}\alpha_1 - P^{*T}\mu_1) + (\epsilon - 1)e_g^T \mu_1 \\ s.t. \ & 0 \leq \alpha_1 \leq c_1, \quad 0 \leq \mu_1 \leq c_u, \end{aligned} \tag{3.24}$$

$$\begin{aligned}\max_{\alpha_2, \mu_2} \ & e_1^T \alpha_2 - \frac{1}{2}(\alpha_2^T E + \mu_2^T P)(F^T F)^{-1}(E^T \alpha_2 + P^T \mu_2) + (1 - \epsilon)e_d^T \mu_2 \\ s.t. \ & 0 \leq \alpha_2 \leq c_2, \quad 0 \leq \mu_2 \leq c_u, \end{aligned} \tag{3.25}$$

where $E = [K(X_1, D^T)\ e_1]$, $F^* = [K(X_2^*, D^T)\ e_1], F = [K(X_2, D^T)\ e_2], P^* =$

$[K(U^*, D^T)\ e_g], P = [K(U, D^T)\ e_d]$, and $\alpha_1, \alpha_2, \mu_1, \mu_2$ are the vectors of Lagrange multipliers.

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 (3.26) and (3.27):

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(E^T E + \sigma I)^{-1}(F^{*T}\alpha_1 - P^{*T}\mu_1), \tag{3.26}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (F^T F + \sigma I)^{-1}(E^T\alpha_2 + P^T\mu_2), \tag{3.27}$$

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 using Eq. (2.10). The algorithm for RUTSVM-CIL is briefly described in Alg. 3.1.

Algorithm 3.1 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.

3.2.4 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 [182] 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 3.2.4.1, and oversampling of universum samples in 3.2.4.2, to create a balance situation for classification. The proposed scheme for the construction of twin hyperplanes in our RUTSVM-CIL is discussed below:

3.2.4.1 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 [179–181] 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 subsection 3.2.6.1(V).

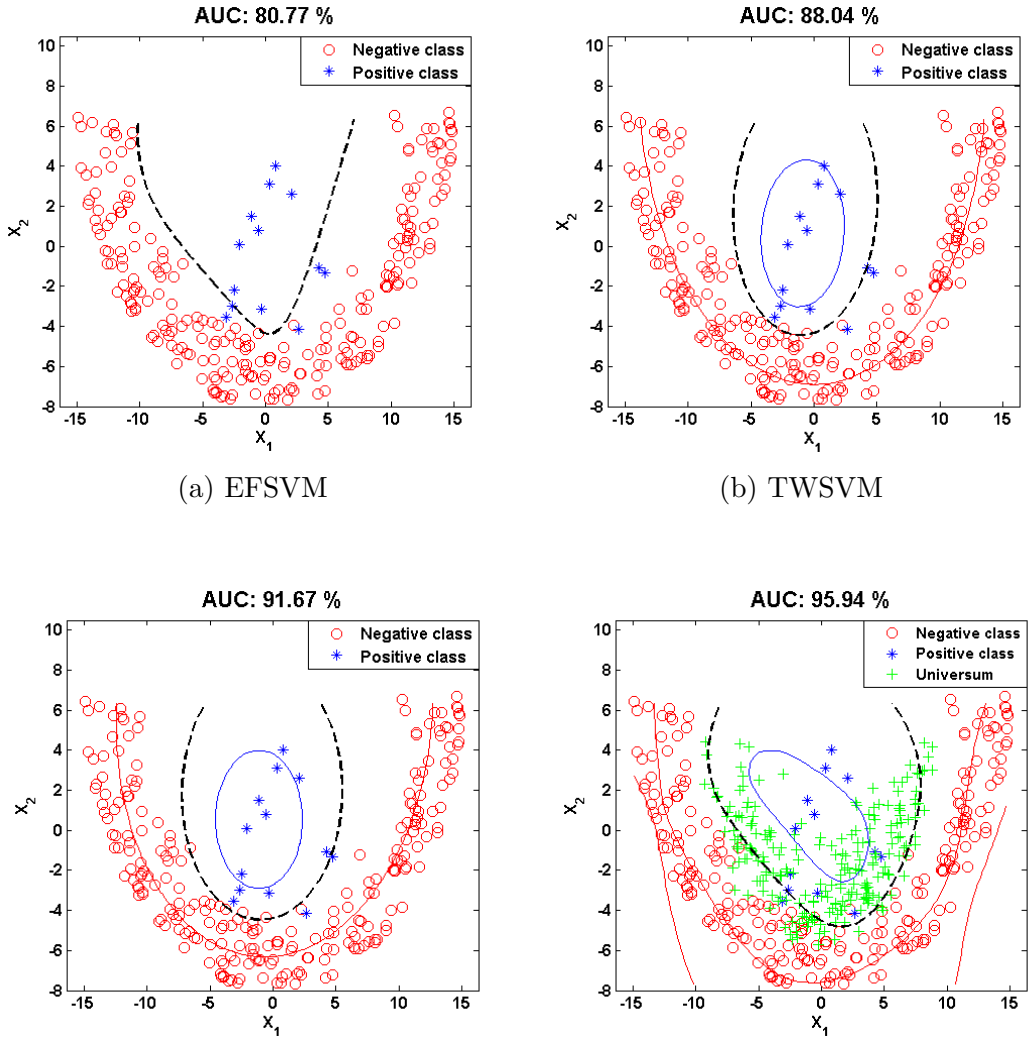
3.2.4.2 Negative (majority) class hyperplane

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 $(1 - \epsilon)$ from the classifier in QPP (3.17), 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. 3.6 and 3.7 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.

3.2.4.3 Kernel matrix

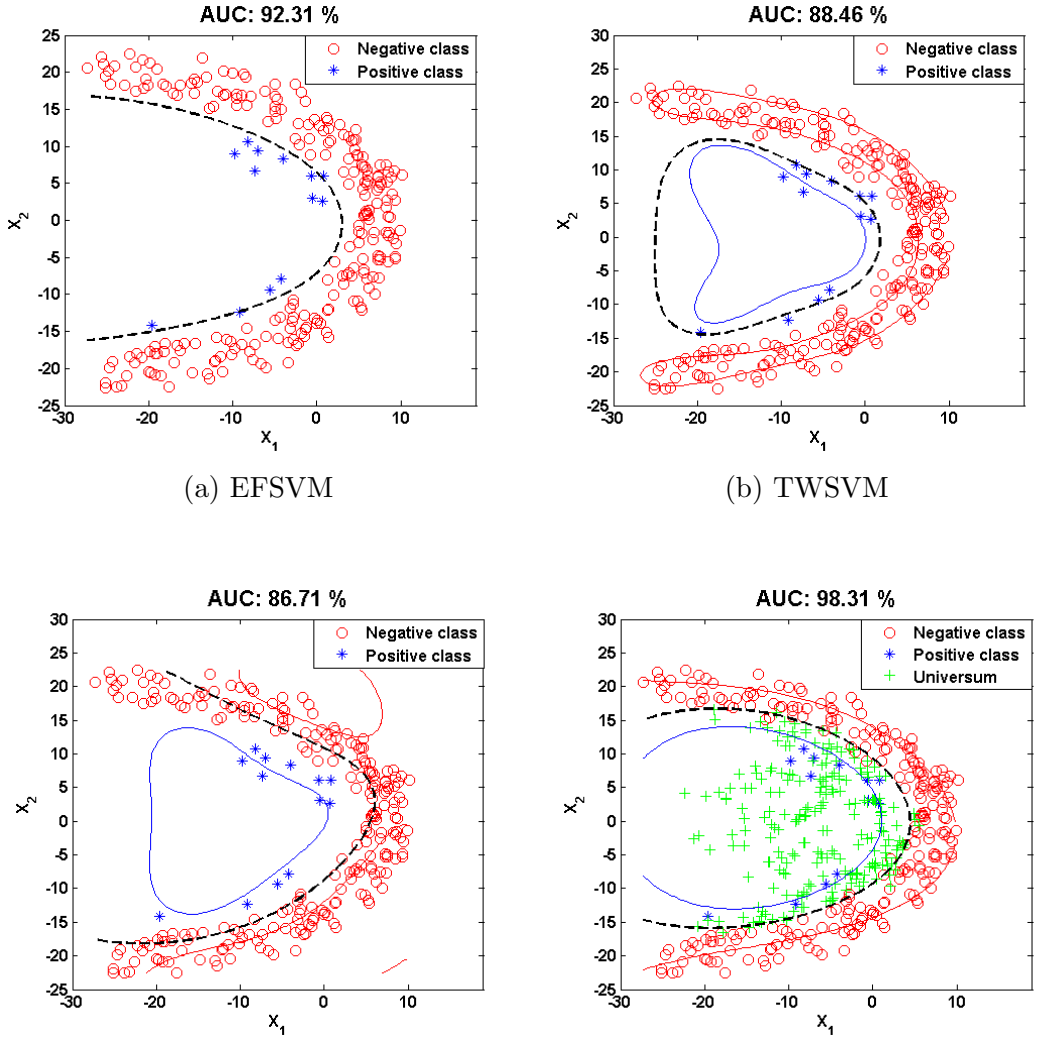
To incorporate the concept of undersampling in the kernel matrix, we choose the rectangular kernel, where the size of the kernel depends on the size of the minority class. The columns of the kernel matrix are equal to twice the size of the minority class, which includes the positive class of size r, 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 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



(a) EFSVM (b) TWSVM

(c) TWSVM-RUS (d) RUTSVM-CIL

Figure 3.6: Performance of EFSVM, TWSVM, TWSVM-RUS and proposed RUTSVM-CIL for the classification of synthetic dataset Crescent_&_full_moon using RBF kernel.



(a) EFSVM (b) TWSVM

(c) TWSVM-RUS (d) RUTSVM-CIL

Figure 3.7: Performance of EFSVM, TWSVM, TWSVM-RUS and proposed RUTSVM-CIL for the classification of synthetic dataset Half_kernel using RBF kernel.

subsection 3.2.6.1(IV). The proposed RUTSVM-CIL also requires less memory to store the kernel matrices, and is suitable for handling large scale imbalanced datasets.

3.2.5 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 case of TWSVM, the time complexity is given as follows [12]:

$$\begin{aligned} T =& O(m_2^3) + O(m_1^3), \\ T =& O(IR \times m_1)^3 + O(m_1^3), \\ T =& O(IR^3 + 1)O(m_1^3). \end{aligned} \tag{3.28}$$

If the imbalance ratio (IR) is equal to 1, then $T = 2 \times O(m/2)^3$ as stated in [12].

In the proposed RUTSVM-CIL, the computational complexity is given by considering the constraints in both QPPs (3.16 and 3.17) as follows:

$$\begin{aligned} T =& O(m_1 + m_1/2)^3 + O(m_1 + (m_2 - m_1))^3, \\ T =& O(m_1 + m_1/2)^3 + O(m_2^3), \\ T =& O(IR^3 + 3.375)O(m_1^3). \end{aligned} \tag{3.29}$$

The time complexity of the proposed RUTSVM-CIL is comparable to TWSVM (Eq. 3.28). 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 classes. The computation complexity of calculating the kernel matrices in TWSVM is

given as follows:

$$\begin{aligned} T =& O(m_1 \times m) + O(m_2 \times m), \\ T =& O(m_1 \times (m_1 + IR \times m_1)) + O(IR \times m_1 \times (m_1 + IR \times m_1)), \\ T =& O(1 + 2IR + IR^2)O(m_1^2). \end{aligned} \tag{3.30}$$

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

$$\begin{aligned} T =& O(m_1 \times 2m_1) + O(m_2 \times 2m_1) + O((m_2 - m_1) \times 2m_1), \\ T =& O(2m_1^2) + O(IR \times m_1 \times 2m_1) + O((IR \times m_1 - m_1) \times 2m_1), \\ T =& O(1 + IR + IR - 1) \times O(2m_1^2), \\ T =& (4IR)O(m_1^2). \end{aligned} \tag{3.31}$$

It is evident from Eqs. (3.30) and (3.31) 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.

3.2.6 Experimental results

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 [55], SVM-RUS [179–181], TWSVM [5], TWSVM-RUS [179–181], TWSVM-SMOTE [183], MMTSSVM [184], FTSVM [175, 185] and UTSVM [27] 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.

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, RBF kernel is used for the kernel function in all the algorithms. The results for all the algorithms are calculated for the non-linear case using RBF kernel. For comparison of classification accuracy of the methods, area under receiver operating characteristics (ROC) curve i.e. AUC is used.

The range of penalty parameter c, and kernel parameter μ is same as in subsection 3.1.5 for all the algorithms. For EFSVM and UTSVM also, the settings are similar as before. 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. For MMTSSVM and FTSVM, $\nu_1 = \nu_2$ is selected from the set $\{0.1, 0.2, \ldots, 0.9\}$. 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.

I. Datasets:

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 [3], and Crossplane (XOR) dataset [178] as shown in Figs. 3.6, 3.7 and 3.2 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. For Crossplane dataset, the parameters k_1 and b_1 [178] 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 3.6, and for Crossplane datasets in Table 3.7.

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 [171] and UCI repository [170]. 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].

The large scale datasets are generated using the approach as mentioned in [186], and the parameters are set as $c_1 = c_2 = c_u = 1$, $\mu = 2$, and $\epsilon = 0.5$. For FTSVM, $\nu_1 = \nu_2$ is set as 0.3. The datasets are made class imbalanced by randomly removing samples of one class.

3.2.6.1 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 Cross-plane.

I. Effect of prior information:

Figs. 3.6 and 3.7 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 mis-classification of data. Moreover, it can be seen in Figs. 3.6 and 3.7 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.

II. 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 3.6 in terms of AUC values and training time. The corresponding average ranks are shown 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. Fig. 3.8 shows the plot of AUC and training time of Crescent_&_full_moon and Half_kernel dataset for different training sizes.

One can observe in Figs. 3.8(a) and 3.8(c) 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. 3.8(c) with 70% training data. This may be attributed to the selection of universum using random averaging of data points. Figs. 3.8(b) and 3.8(d) 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.

III. Crossplane dataset:

Numerical experiments are performed on synthetic Crossplane dataset to verify the effectiveness of the proposed RUTSVM-CIL. Table 3.7 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 3.7.

IV. Effect of imbalance ratio (IR):

To verify the efficacy of the reduced kernel, the performance comparison is made using Crossplane dataset in Fig. 3.9. It is clear from Fig. 3.9 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. 3.9(c) 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

Table 3.6: AUC (%) values and training time of the algorithms for classification of synthetic class imbalanced datasets ($IR = 19$) with different training size. The average ranks are calculated on the basis of AUC. Bold values indicate highest AUC.

Dataset (Train size, Test size)	**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 (c, μ, K) Time (s)	**UTSVM** AUC (c, μ, ϵ) Time (s)	**Proposed RUTSVM-CIL** AUC (c, μ, ϵ) Time (s)
Crescent_&_full_moon $(150 \times 2, 350 \times 2)$	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 $(200 \times 2, 300 \times 2)$	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 $(250 \times 2, 250 \times 2)$	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 $(350 \times 2, 150 \times 2)$	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 $(150 \times 2, 350 \times 2)$	20.43	**97.2222** $(10^2, 2^4, 3)$ 0.2645	87.8012 $(10^3, 2^5)$ 0.0041	58.0321 $(10^{-4}, 2^5)$ 0.0321	84.5884 $(10^{-2}, 2^4)$ 0.0061	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 $(200 \times 2, 300 \times 2)$	19	93.3333 $(10^3, 2^5, 3)$ 0.4644	96.1404 $(10^2, 2^5)$ 0.0063	93.3333 $(10^{-2}, 2^4)$ 0.0552	86.6667 $(10^{-2}, 2^5)$ 0.0064	**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 $(250 \times 2, 250 \times 2)$	19.83	92.3077 $(10^2, 2^4, 9)$ 0.7261	94.7257 $(10^1, 2^4)$ 0.0078	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 $(350 \times 2, 150 \times 2)$	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
Average AUC		89.8797	91.2378	86.2891	88.7062	90.0082	80.7991	87.9312	88.4975	**93.9681**
Average rank		4.875	3.125	6.125	5.6875	4.625	6.5	5.4375	6.1875	**2.4375**

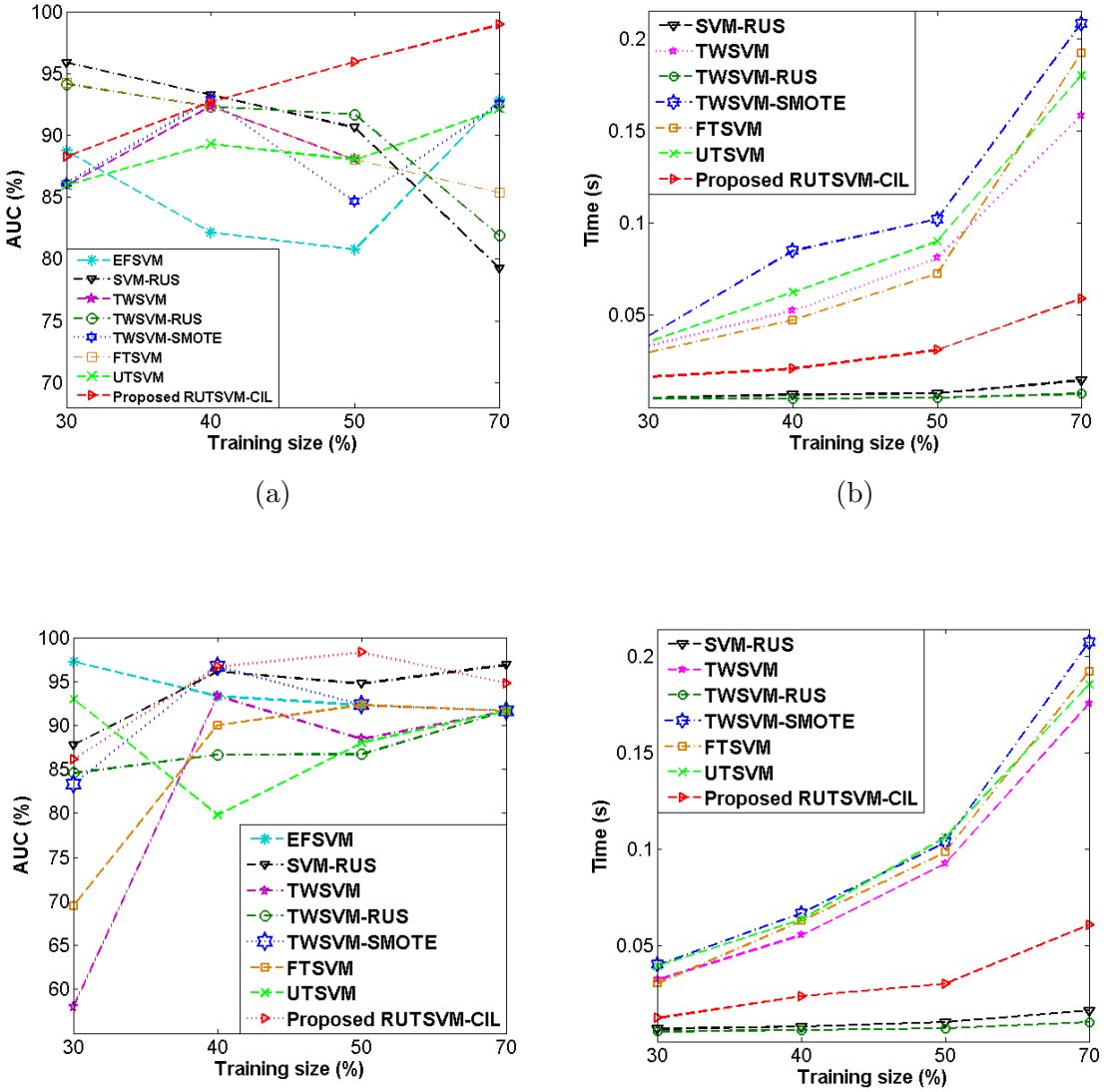


Figure 3.8: Plot of AUC vs training size, and time vs training size for Crescent_&_full_moon in (a) and (b), and Half_kernel in (c) and (d) respectively. Number of samples are 500.

Table 3.7: AUC (%) and training time of proposed RUTSVM-CIL and existing algorithms for classification of synthetic Crossplane imbalanced datasets. Average rank is calculated on AUC.

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 (c, μ, K) Time (s)	**UTSVM** AUC (c, μ, ϵ) Time (s)	**Proposed RUTSVM-CIL** AUC (c, μ, ϵ) Time (s)
Crossplane_120 $(50 \times 2, 70 \times 2)$	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 $(0.1, 2^{-5})$ 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 $(150 \times 2, 100 \times 2)$	4	93.1818 $(10^{-5}, 2^{-5}, 3)$ 0.2645	95.5128 $(10^{-5}, 2^{-5})$ 0.0323	94.1725 $(10^{-2}, 2^{2})$ 0.0327	95.1632 $(10^{-5}, 2^{1})$ 0.0079	98.7179 $(10^{-5}, 2^{-5})$ 0.0686	75.4079 $(0.1, 2^{-1})$ 0.029	95.4545 $(0.3, 2^{-1})$ 0.0326	97.0862 $(10^{-2}, 2^{-4}, 0.1)$ 0.0359	**99.359** $(10^{-4}, 2^{-4}, 0.6)$ 0.0342
Crossplane_330 $(110 \times 2, 220 \times 2)$	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 $(0.1, 2^{-5})$ 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 $(133 \times 2, 267 \times 2)$	3	95.2381 $(10^{0}, 2^{-5}, 3)$ 0.2086	98.4711 $(10^{1}, 2^{-5})$ 0.0552	97.3739 $(10^{-4}, 2^{-2})$ 0.0246	**99.5098** $(10^{-5}, 2^{-4})$ 0.0104	98.9613 $(10^{-4}, 2^{-2})$ 0.0741	51.3539 $(0.1, 2^{-4})$ 0.0229	**99.5098** $(0.3, 2^{-2})$ 0.026	98.9613 $(10^{-2}, 2^{-2}, 0.2)$ 0.028	98.4127 $(10^{-3}, 2^{-2}, 0.6)$ 0.0339
Average rank		7.75	6.25	5.625	3.625	3.125	7.25	4.875	3.75	**2.75**

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 Figs. 3.9(c) and 3.9(d).

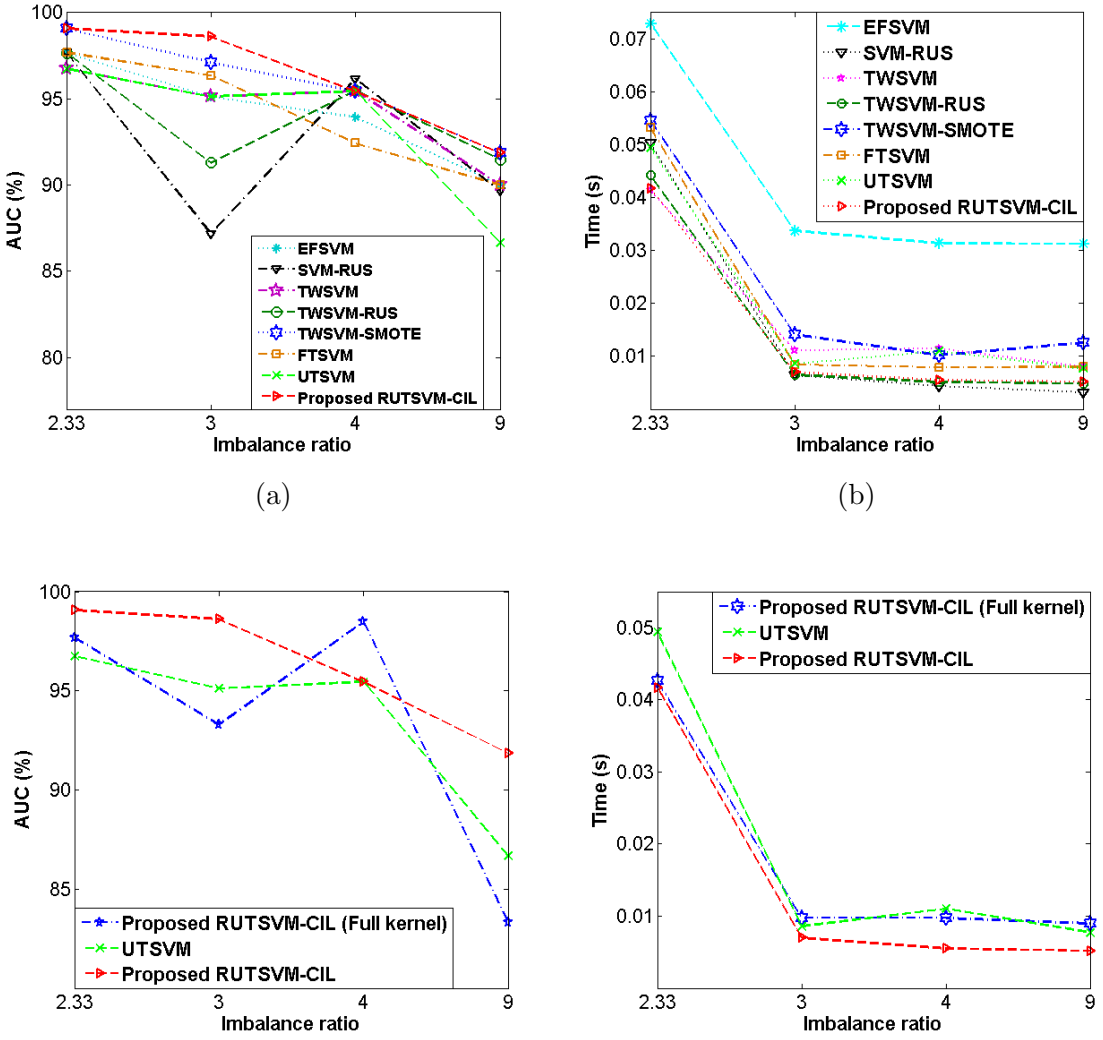


Figure 3.9: Plot of AUC and time vs imbalance ratio (IR) is shown in (a) and (b) respectively for Crossplane dataset containing 200 data points. A comparison with proposed RUTSVM-CIL with full kernel is shown in (c) and (d).

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.

V. 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 $(r/2)$, 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. 3.10, 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^*.

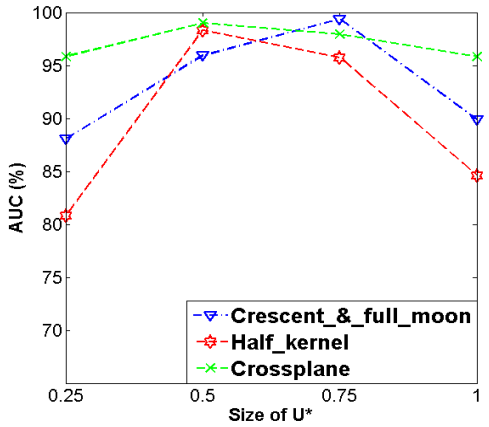


Figure 3.10: Plot of AUC vs size of U^* on synthetic datasets. The size of U^* is represented as a fraction of the minority class samples.

3.2.6.2 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.

I. Generalization performance:

The classification accuracy of the algorithms is shown in Table 3.8 in terms of AUC values and training time, with the corresponding average ranks. 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. 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 3.8 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 the proposed RUTSVM-CIL with existing algorithms on different sets of testing data, we calculated mean and standard deviation (SD) of AUC and G-mean [187] in Table 3.9. The AUC and G-mean are calculated on 5 folds of testing data. As shown in Table 3.9, 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. 3.11 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. 3.11 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

Table 3.8: Comparison of proposed RUTSVM-CIL on AUC (%) and training time with existing algorithms for classification on real world imbalanced datasets. Rank is based on AUC.

Dataset (Train size, Test size)	**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 (c, μ, K) 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 $(120 \times 7, 124 \times 7)$	9.17	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 $(120 \times 6, 120 \times 6)$	11	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 $(150 \times 6, 182 \times 6)$	12.28	**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 $(100 \times 7, 105 \times 7)$	9.25	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 $(100 \times 6, 103 \times 6)$	9.15	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 $(110 \times 6, 110 \times 6)$	10	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 $(100 \times 9, 105 \times 9)$	11.06	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 $(80 \times 9, 92 \times 9)$	9.12	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 $(900 \times 9, 929 \times 9)$	13.87	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

Table 3.8 (contd.)

Dataset (Train size, Test size)	**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 (c, μ, K) Time (s)	**UTSVM** AUC (c, μ, ϵ) Time (s)	**Proposed RUTSVM-CIL** AUC (c, μ, ϵ) Time (s)
Yeast-0-2-5-6_vs_3-7-8-9 $(500 \times 8, 504 \times 8)$	9.14	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 $(500 \times 8, 504 \times 8)$	9.14	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 $(250 \times 8, 256 \times 8)$	9.12	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 $(150 \times 6, 130 \times 6)$	13	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 $(200 \times 3, 106 \times 3)$	2.78	**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
Glass4 $(150 \times 9, 64 \times 9)$	15.46	**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 $(150 \times 7, 186 \times 7)$	8.6	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 $(350 \times 7, 381 \times 7)$	16.4	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 $(400 \times 18, 446 \times 18)$	2.9	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

Table 3.8 (contd.)

Dataset (Train size, Test size)	**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 (c, μ, K) Time (s)	**UTSVM** AUC (c, μ, ϵ) Time (s)	**Proposed RUTSVM-CIL** AUC (c, μ, ϵ) Time (s)
Vehicle2 $(400 \times 18, 446 \times 18)$	2.88	**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 $(500 \times 8, 984 \times 8)$	8.1	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 $(500 \times 8, 2468 \times 8)$	2.46	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 $(200 \times 8, 259 \times 8)$	14.3	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 $(250 \times 8, 264 \times 8)$	9.08	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 $(250 \times 8, 233 \times 8)$	23.15	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 $(180 \times 7, 131 \times 7)$	4.76	**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 $(500 \times 8, 984 \times 8)$	32.73	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 $(100 \times 33, 94 \times 33)$	3.22	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 $(90 \times 5, 125 \times 5)$	5.14	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 AUC		79.2743	80.68	79.3939	79.3884	78.785	68.5603	78.9051	78.7583	**82.7742**
Average rank		4.7857	4.9821	4.6429	4.625	5.5536	7.2321	5.1429	5.375	**2.6607**

Table 3.9: Comparison of RUTSVM-CIL with existing algorithms on mean and standard deviation (SD) of AUC (%) and G-mean (%) for classification of imbalanced datasets.

Dataset (Train size, Test size)	**IR** (All samples)	**EFSVM** AUC G-mean	**SVM_RUS** AUC G-mean	**TWSVM** AUC G-mean	**TWSVM_RUS** AUC G-mean	**TWSVM-SMOTE** AUC G-mean	**MMTSSVM** AUC G-mean	**FTSVM** AUC G-mean	**UTSVM** AUC G-mean	**Proposed RUTSVM-CIL** AUC G-mean
Ecoli-0-1_vs_2-3-5 (120×7, 124×7)	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 (120×6, 120×6)	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 (150×6, 182×6)	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 (100×7, 105×7)	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 (100×6, 103×6)	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 (110×6, 110×6)	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 (100×9, 105×9)	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 (80×9, 92×9)	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 (900×9, 929×9)	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 (500×8, 504×8)	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 (500×8, 504×8)	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 (250×8, 256×8)	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 (150×6, 130×6)	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 (200×3, 106×3)	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

Table 3.9 (contd.)

Dataset (Train size, Test size)	**IR** (All samples)	**EFSVM** AUC G-mean	**SVM_RUS** AUC G-mean	**TWSVM** AUC G-mean	**TWSVM_RUS** AUC G-mean	**TWSVM-SMOTE** AUC G-mean	**MMTSSVM** AUC G-mean	**FTSVM** AUC G-mean	**UTSVM** AUC G-mean	**Proposed RUTSVM-CIL** AUC G-mean
Glass4 (150×9, 64×9)	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 (150×7, 186×7)	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 (350×7, 381×7)	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
Vehicle 1 (400×18, 446×18)	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 (400×18, 446×18)	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 (500×8, 984×8)	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 (500×8, 2468×8)	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 (200×8, 259×8)	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 (250×8, 264×8)	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 (250×8, 233×8)	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 (180×7, 131×7)	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 (500×8, 984×8)	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 (100×33, 94×33)	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 (90×5, 125×5)	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**

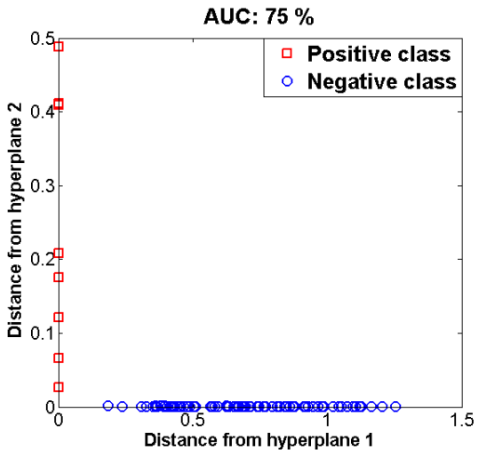
Fig. 3.11(c) that the distance of negative class from its hyperplane is more in case of RUTSVM-CIL in comparison to TWSVM and UTSVM in Figs. 3.11(a) and 3.11(b) respectively. This shows the effect of universum in reducing the bias to the negative or majority class. The majority class data points which are near to the minority class are given less importance.

II. Computation time:

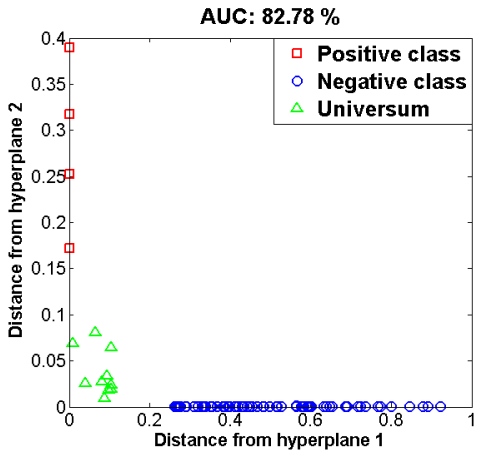
One can observe the computation time of the different algorithms in Table 3.8. The training time of RUTSVM-CIL is less than all the existing algorithms except SVM-RUS and TWSVM-RUS. This is due to undersampling of the majority class in one QPP, as well as oversampling of the universum in the other QPP of RUTSVM-CIL. Also, RUTSVM-CIL finds the solution of two smaller sized QPPs with reduced kernel matrix. It can be seen in Table 3.8 that the training time of UTSVM is more than TWSVM due to the universum. However, in comparison to TWSVM, our RUTSVM-CIL takes lesser time while incorporating the universum data. It is visible in Table 3.8 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.

3.2.6.3 Statistical tests

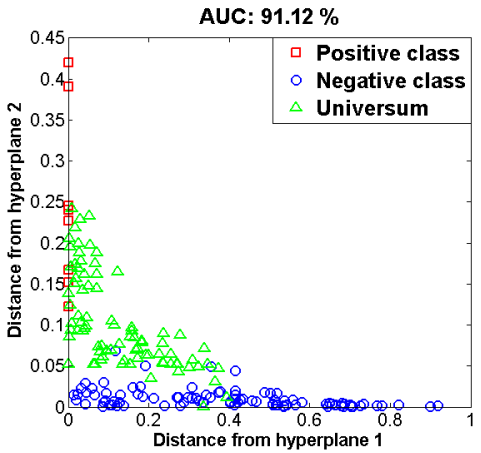
Similar to section (3.1.5.2), we apply the Friedman test to check the statistical difference for the 9 algorithms on 28 class imbalanced datasets. First, we assume the null hypothesis as there is no difference between the methods. The χ_F^2 value for Friedman statistic is calculated using average ranks from Table 3.8. By applying the formula (2.54), we get $\chi_F^2 = 41.9496$.



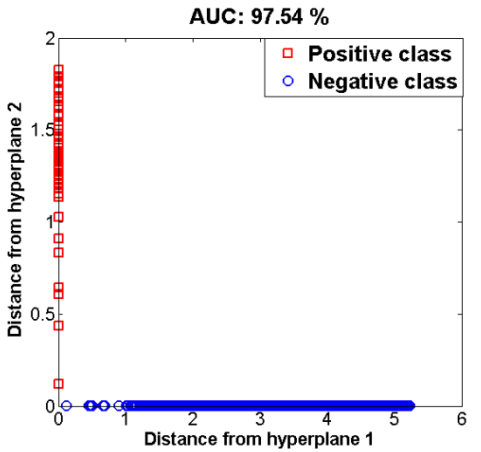
(a) TWSVM



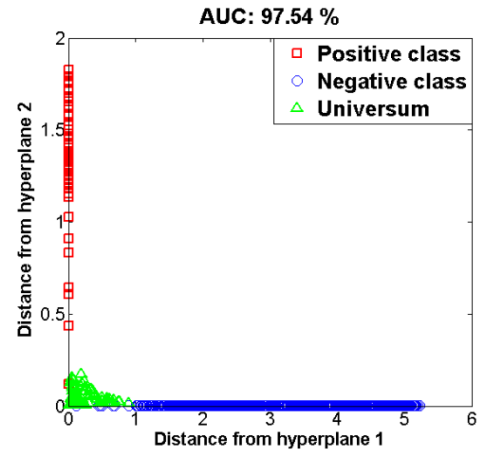
(b) UTSVM



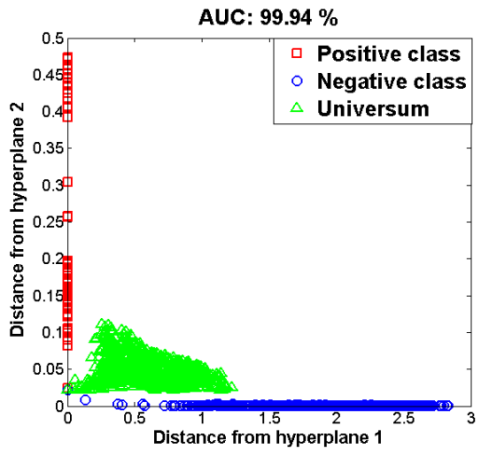
(c) Proposed RUTSVM-CIL



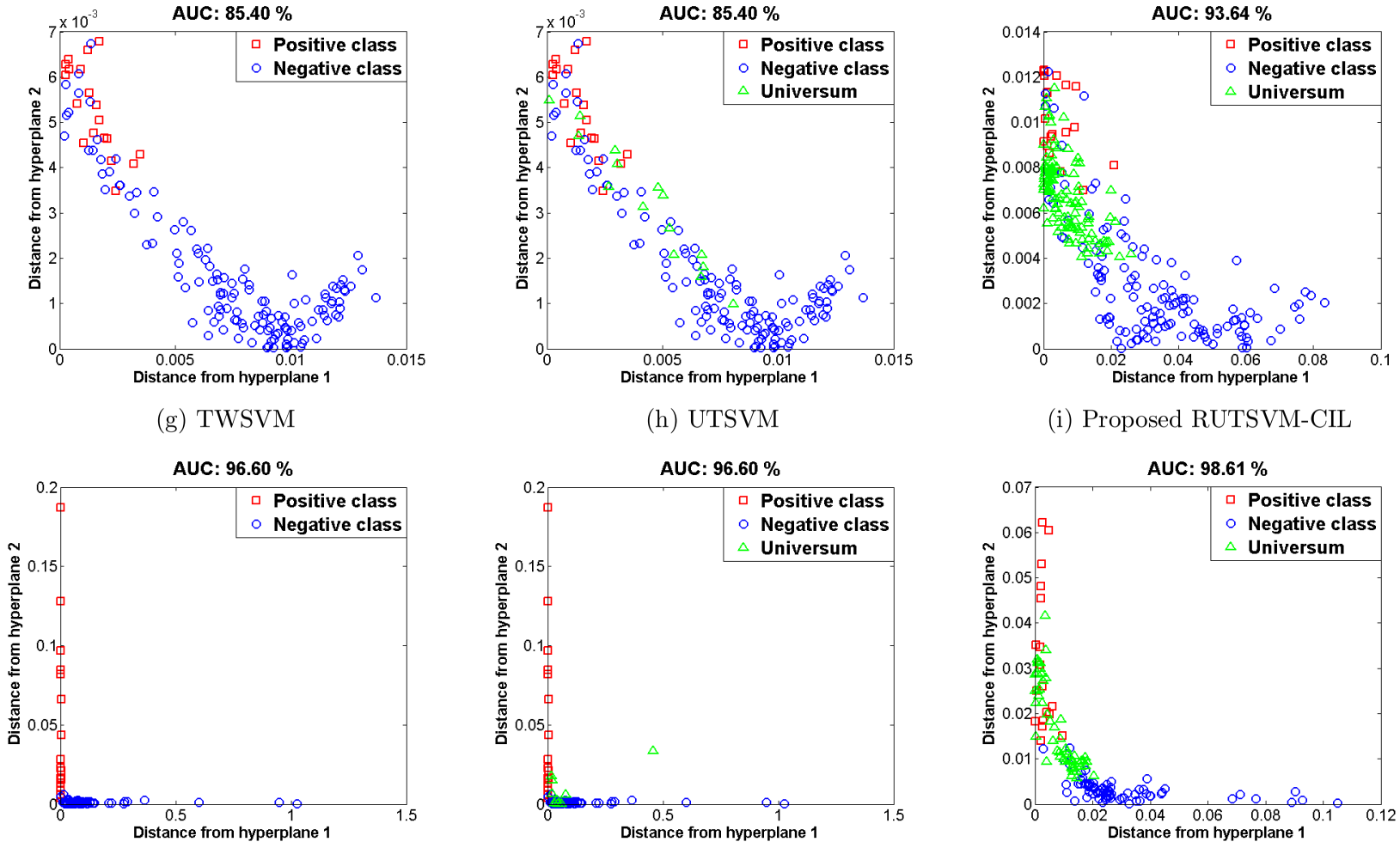
(d) TWSVM



(e) UTSVM



(f) Proposed RUTSVM-CIL



(g) TWSVM (h) UTSVM (i) Proposed RUTSVM-CIL

(j) TWSVM (k) UTSVM (l) Proposed RUTSVM-CIL

Figure 3.11: Plot of distance of data points from the classifying hyperplanes for (a-c) Ecoli-0-4-6_vs_5, (d-f) Shuttle-c0-vs-c4, (g-i) Ecoli3, and (j-l) New_thyroid2 datasets.

The F_F value is calculated as

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

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

Now, to check the pairwise difference between the proposed and existing algorithms, we use the Nemenyi posthoc test using Eq. (2.56). For significant pairwise difference between the methods at significance level of $\alpha = 0.10$, the average ranks of the methods shown in Table 3.8 should differ by atleast $2.855\sqrt{\frac{9(9+1)}{6\times 28}} = 2.0896$. The pairwise difference between the methods is shown in Table 3.10.

Table 3.10: Pairwise significant difference of proposed RFLSTSVM-CIL with existing algorithms.

Significance	**EFSVM**	**SVM-RUS**	**TWSVM**	**TWSVM-RUS**	**TWSVM-SMOTE**	**MMTSSVM**	**FTSVM**	**UTSVM**
Proposed RUTSVM-CIL	**Yes**	**Yes**	No	No	**Yes**	**Yes**	**Yes**	**Yes**

3.2.6.4 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 [186] datasets using RBF kernel. Table 3.11 shows the performance comparison on large scale imbalanced datasets, where the training time is shown in seconds. One can observe from Table 3.11 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

Table 3.11: Comparison of proposed RUTSVM-CIL on AUC (%) and training time with existing algorithms for classification on large scale imbalanced datasets.

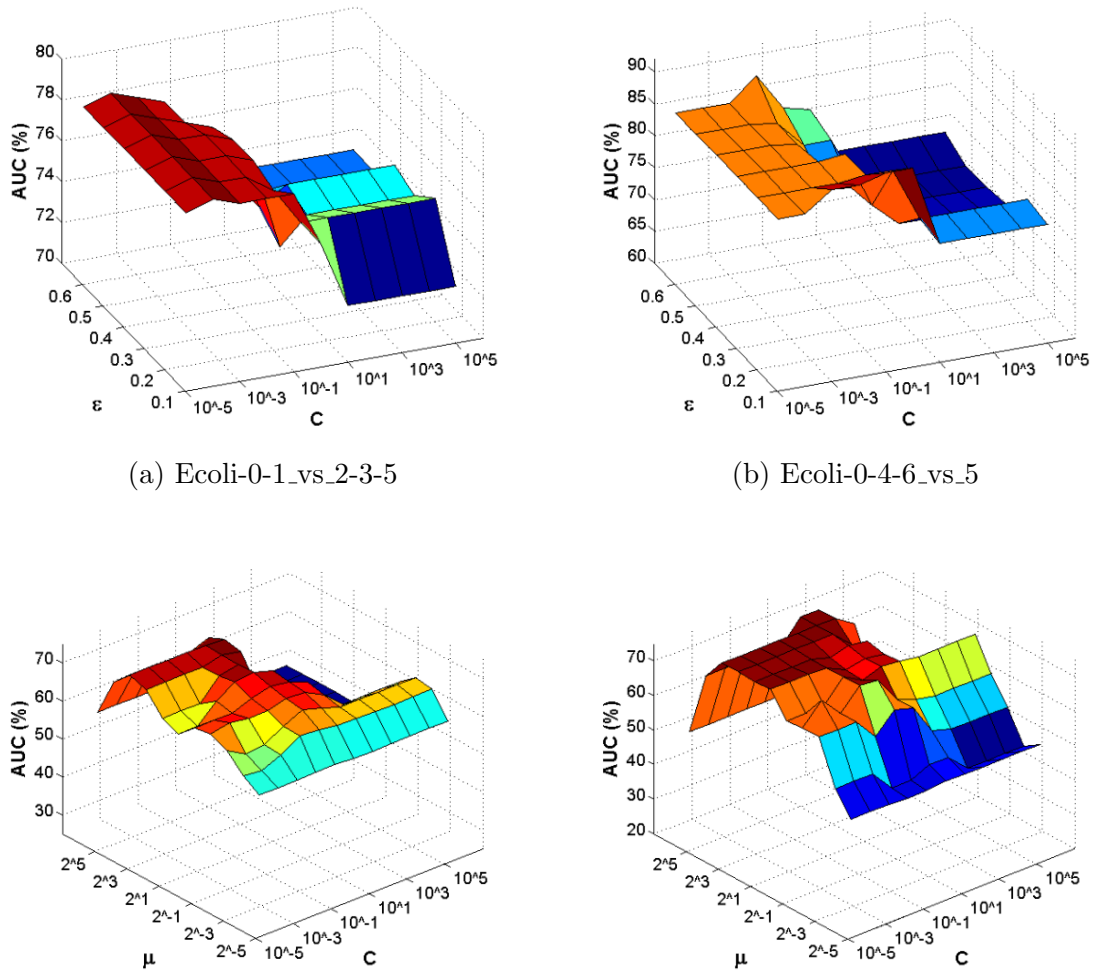
Dataset (Train size, Test size)	**IR** (All samples)	**TWSVM** AUC Time (s)	**FTSVM** AUC Time (s)	**UTSVM** AUC Time (s)	**Proposed RUTSVM-CIL** AUC Time (s)
NDC-1 (3336×10, 335×10)	5.67	96.1781 13.0017	95.4849 14.3013	96.3524 15.5044	**99.3031** 7.0055
NDC-2 (5008×10, 502×10)	5.68	**97.8261** 31.9098	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

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.

3.2.6.5 Insensitivity analysis

In Fig. 3.12, 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 Figs. 3.12(a) and 3.12(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.



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

(b) Ecoli-0-4-6_vs_5

(c) Yeast1

(d) Yeast2vs8

Figure 3.12: Insensitivity performance of proposed RUTSVM-CIL for classification of real world imbalance datasets to the user specified parameters using RBF kernel.

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 Figs. 3.12(c) and 3.12(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.

3.3 Summary

In this chapter, we proposed two novel SVM based algorithms for class imbalance learning viz. (which are) RFLSTSVM-CIL and RUTSVM-CIL. We also proposed a novel fuzzy membership function specifically for class imbalance learning, which gives different range of fuzzy membership values for different datasets. The different range of the fuzzy membership function helps in giving proper weights to the data points in different imbalance scenarios. The proposed RFLSTSVM-CIL has shown good generalization performance with less training time in comparison to the existing algorithms on noisy datasets.

The proposed RUTSVM-CIL is a novel computationally efficient model 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. RUTSVM-CIL has shown good generalization performance with less training time on several synthetic and real world datasets.

In the next chapter, we focus on algorithms based on universum learning for feature extraction and classification in biomedical datasets related to brain disorders.

Chapter 4

Universum learning for neurological disorders

The previous chapter discussed a universum based technique for class imbalance learning. Universum selection depends on the type of problem. Therefore, we present techniques for applying universum based algorithms on problems such as detection of diseases. In this chapter, we present two novel universum based techniques, and apply on brain disorder datasets. Section 4.1 explains the proposed universum based technique[1] for detection of epilepsy, while section 4.2 presents a universum based feature elimination algorithm[2] for diagnosis of Alzheimer's disease.

In the following section, we present a universum based technique for detection of epilepsy using electroencephalogram (EEG) signals.

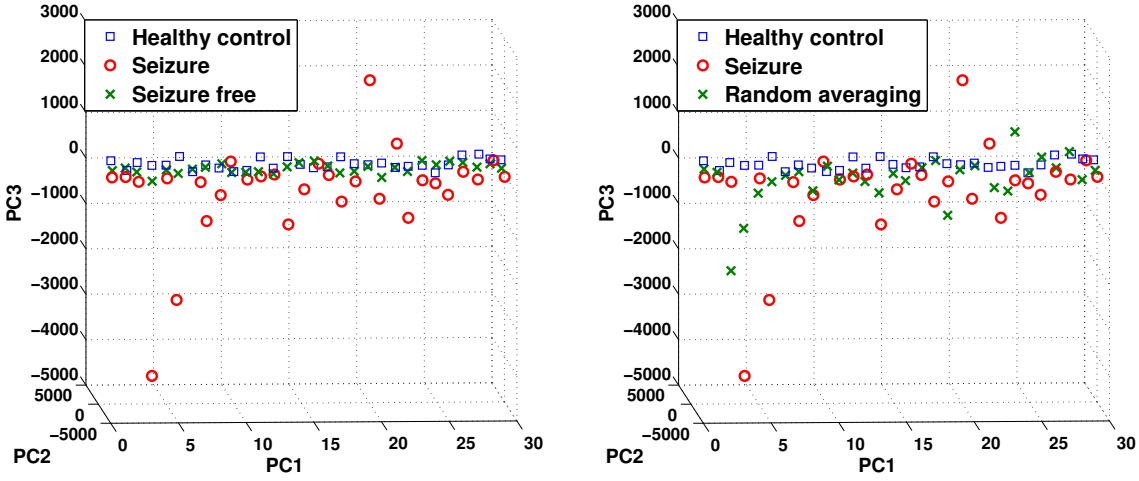
[1] **B. Richhariya**, M. Tanveer. EEG signal classification using universum support vector machine. *Expert Systems with Applications*, Elsevier, 106:169-182, 2018, DOI: `https://doi.org/10.1016/j.eswa.2018.03.053`.
[**SCI Indexed Impact Factor: 6.954**]

[2] **B. Richhariya**, M. Tanveer, A.H. Rashid, Alzheimer's Disease Neuroimaging Initiative. Diagnosis of Alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE). *Biomedical Signal Processing and Control*, Elsevier, 59:101903, 2020, DOI: `https://doi.org/10.1016/j.bspc.2020.101903`.
[**SCI Indexed Impact Factor: 3.880**]

4.1 EEG signal classification using universum support vector machine

The major challenge with universum based approach is the proper selection of universum data points. In digit classification problem [10], the universum data is selected based on similarity of digits. For example, digit '3' is chosen as universum for classifying '5' and '8' since its shape is similar to both '5' and '8'. Chapelle et al. [19] presented an analysis for the selection of proper universum data. Universum samples are generated for classification of faces [188] using the random averaging approach, where the average of the pixels of two faces is used as the universum. An in-between-universum (IBU) approach is also proposed [189] for the proper selection of universum. The practical conditions for choosing the universum data are given in [190, 191]. To reduce the training time of TWSVM, twin support vector machine (TWSVM) [12] was proposed where two quadratic programming problems (QPPs) of smaller size are solved to obtain the classifier. For the classification of seizure EEG signals, for the first time TWSVM is used in this work. Qi et al. [27] proposed a universum twin support vector machine (UTSVM) to reduce the computational complexity of USVM and used the random averaging approach for universum selection. Xu et al. [71] also used the random averaging scheme for selecting the universum data. Since the random averaging approach suffers from the effect of outliers, the method of generation of universum data depends solely on the type of application and is currently an area of research.

Motivated by the work on universum support vector machine in [22, 56, 57], we propose a novel approach of selecting the universum in the classification of EEG signals for seizure detection. Since universum based support vector machines have not been used for the classification of EEG signals, we also present an application of USVM and UTSVM for EEG signals. For the classification of EEG signals in the healthy and seizure (ictal) classes, the interictal EEG signals are chosen as the universum, which corresponds to the EEG recording for the time period in between the seizures in a patient with epilepsy. The proposed approach of EEG classification is tested for



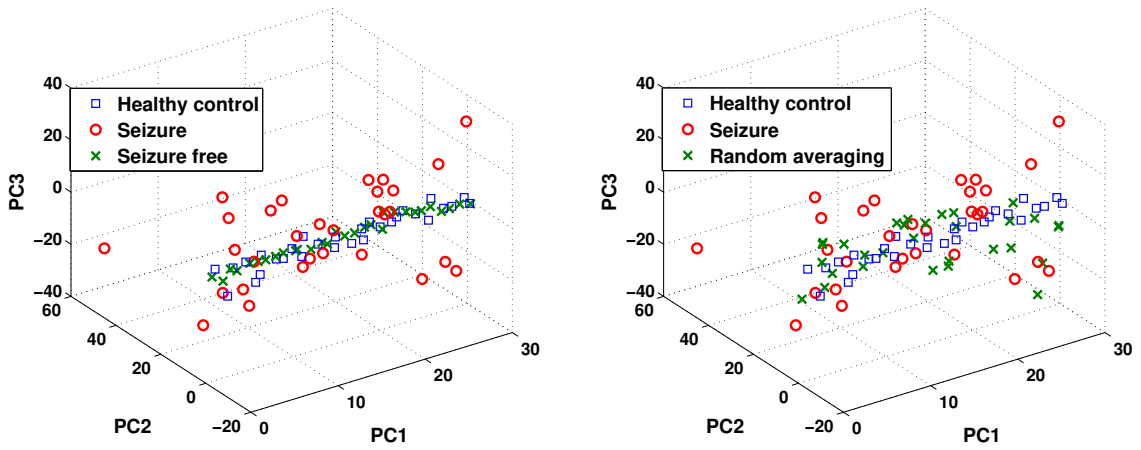
(a) (b)

Figure 4.1: Distribution of data points of set Z set as healthy control, S as seizure using PCA up to 3 principal components in (a) for proposed method i.e., using set N (seizure free) data points as universum and in (b) random averaging is used for generating the universum.

different datasets that are generated using various feature extraction techniques, and the results are compared with existing methods.

4.1.1 Proposed approach using universum

In many classification approaches for EEG signals, the prior information about the distribution of EEG data is not utilized. Due to this, the classification techniques are not able to give better generalization performance, even with the most efficient feature extraction technique. The universum based approach gives some prior information in the construction of the classifier. So, we used a universum based approach with support vector machine to classify the EEG signals. Further, in the datasets generated from the EEG signals, many data points behave as outliers, especially in case of seizure signal as shown in Figs. 4.1 and 4.2. Consequently, the traditional approach of universum based support vector machine based on random averaging [27, 71] is not so efficient in giving the prior information. The outlier data points affect the generation of the universum points in the random averaging approach, which leads to incorrect classification.



(a) (b)

Figure 4.2: Distribution of data points of set Z as healthy control, S as seizure using ICA up to 3 principal components (PCs) in (a) the proposed approach using seizure free data points as universum and (b) universum data points generated using random averaging.

The proposed approach of universum support vector machine (USVM) selects the universum points from the EEG dataset itself. We take the interictal or seizure free signals from the EEG dataset [192] as the universum. Since the variation of the signal in the seizure free state comes in between the variation of healthy and seizure EEG signals, this gives the required prior information to the support vector machine classifier in an effective manner. Moreover, there are no outliers in the universum data since our universum data is not generated from the training data and thus there is no effect of noise. A comparison of the proposed approach with the traditional random averaging scheme is illustrated in Figs. 4.1 and 4.2, where the universum data points of the proposed approach lie in between the two classes.

Further, we use the proposed approach with universum twin support vector machine (UTSVM) which is a more efficient technique in terms of computational complexity. A brief illustration of our methodology is given in Fig. 4.3.

The steps involved in the proposed approach for classification of EEG signals are as follows:

(i). Choose a feature extraction technique and extract the features from the training

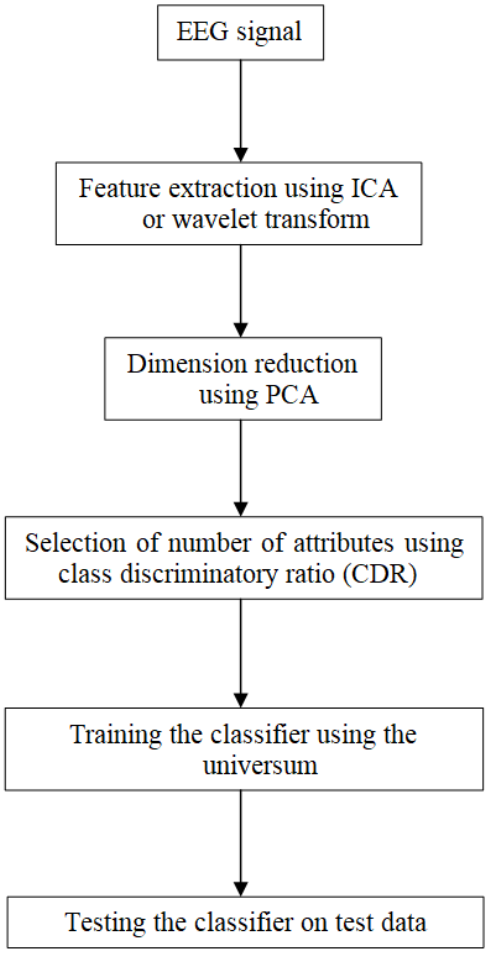


Figure 4.3: Proposed approach using universum.

data consisting of healthy and seizure data points.

(ii). Extract the features from the universum points which are taken from seizure free dataset.

(iii). Reduce the dimension of the feature vector using PCA [193] and class discriminatory ratio (CDR) [194].

(iv). Train the model using training data with the universum.

(v). Test the model using testing data, step (iii) and the classifier.

In this work, different feature extraction techniques are used to extract the appropriate features from the datasets such as principal component analysis (PCA), independent component analysis (ICA) and wavelet transform with different families of wavelet such as db1, db2, db4, db6 and Haar wavelet.

4.1.2 Experimental results

In this section, numerical experiments are performed for the classification of EEG signals of healthy and seizure states. The EEG dataset used in this work is available online [192]. The dataset consists of five sets viz. Z, O, N, F and S. Each set contains 100 single-channel EEG signals sampled at a sampling rate of 173.61 Hz and of 23.6 seconds duration. The sets Z and O are surface EEG recordings of five healthy volunteers with eyes open and closed respectively. The sets N and F are recordings of five patients in the interictal state and the region of recording is the hippocampal formation of the opposite hemisphere of the brain in N and the epileptogenic zone in F. The set S is for the ictal state consisting of seizure recordings from all the recording sites exhibiting ictal activity. The mode of EEG recording is intra-cranial for N, F and S. For all the EEG signals, same 128-channel amplifier system is used with an average common reference.

In the numerical experiments, the training and testing set consists of 50 samples each, chosen from the sets Z, O and S each containing 100 samples. In the proposed approach, the universum is chosen from the set N which contains the interictal EEG signals. For feature extraction, various techniques are applied including principal component analysis (PCA), independent component analysis (ICA) and wavelet transform. In case of wavelet transform, several families of wavelets are applied with different levels of decomposition as used in the available literature. Discrete wavelet transform (DWT) is implemented using different families of wavelet on specific levels of decomposition. The set of the approximation and decomposition coefficients is taken as the feature vector. The level of decomposition is set at level-3 for Daubechies wavelet- db2, db4, and Haar wavelet. For db1 and db6 wavelets, we used level-2 decomposition. In case of ICA and wavelet transform, PCA is applied for the dimension

reduction. The implementation of ICA is same as in [194] (ICA Architecture1). The class discriminatory ratio (CDR) is used to sort the PCA components and to choose the most relevant PCA components. To check the effectiveness, the results of the proposed method for universum are compared with SVM, LSSVM, and USVM with random averaging scheme [27, 71]. In case of UTSVM, we made a comparison with TWSVM and UTSVM with random averaging.

The range of penalty parameter is same as in section 3.1.5 for all the algorithms. For USVM, proposed USVM, UTSVM and proposed UTSVM, the number of universum samples i.e. u is taken from the set $\{10, 20, 30, 40\}$ and ϵ is chosen by varying values from the set $\{0.1, 0.2, \ldots, 0.7\}$. For the selection of the optimal parameters, 5-fold cross-validation is used. In the proposed approaches, universum is selected from the set N of the EEG database and for the existing universum methods random averaging is used for generating the universum data. RBF kernel is used in all the cases, and the value of μ is calculated as per the following formula [195] in all the methods,

$$\mu = \frac{1}{N^2} \sum_{i,j=1}^{N} \|x_i - x_j\|^2, \tag{4.1}$$

where x_i, x_j represent each data point, and N is the total number of data points.

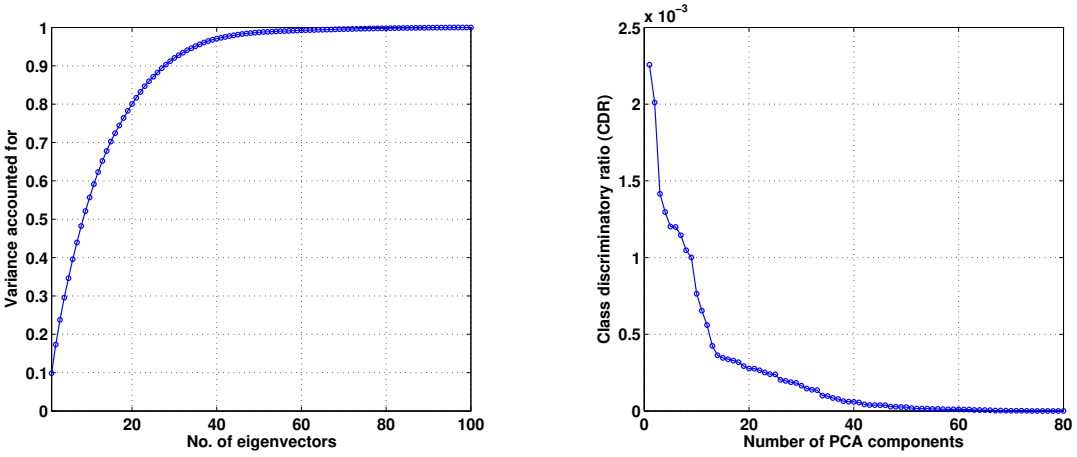
For all the datasets, the number of attributes are decided on the basis of two factors, (a) variance accounted for [193], and (b) class discriminatory ratio (CDR). The approach of calculating CDR of components is taken from [194] as

$$r = \frac{\sigma_{between}}{\sigma_{within}}, \tag{4.2}$$

where $\sigma_{between} = \sum_{i=1}^{C} (\overline{x_i} - \overline{x})^2$ is the variance of C class means, x is mean of all samples, and $\sigma_{within} = \sum_{i=1}^{C} \sum_{j=1}^{C} (x_{ij} - \overline{x_j})^2$ is sum of the within class variance of all the C classes.

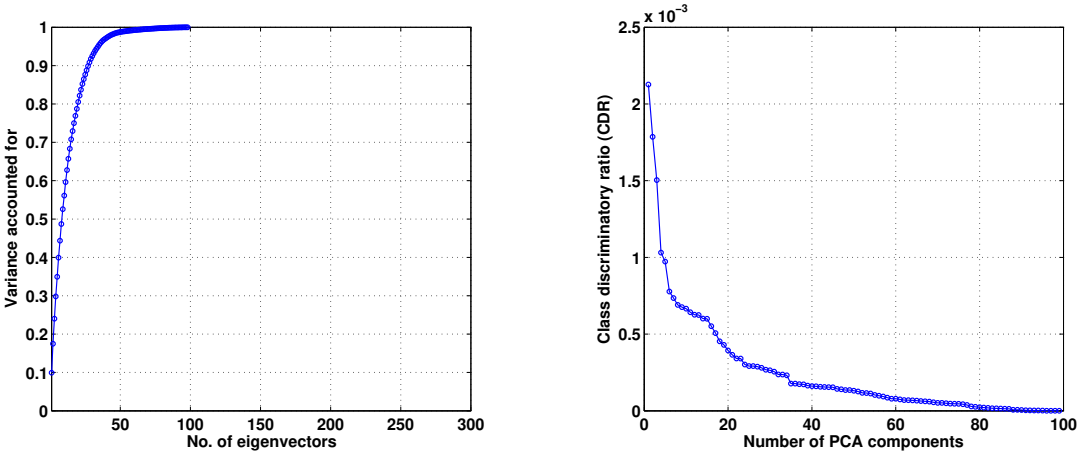
The plots for variance and CDR are shown in Figs. 4.4 and 4.5 for Z & S dataset using PCA and ICA respectively. In Fig. 4.6, the generalization performance of the proposed approach for UTSVM is compared with random averaging approach for Z &

S and O & S using PCA, O & S using ICA, and O & S using wavelet feature extraction technique. The accuracy is shown for different number of universum points. In Fig. 4.6, it can be seen that in all cases the proposed approach is giving higher accuracy in comparison to the traditional approach. Also the effect of outliers is clearly visible in Figs. 4.6(c) and 4.6(d) for the random averaging approach, where the accuracy decreases for some sets of universum. This justifies our selection of the universum.



(a) (b)

Figure 4.4: (a) Variance of data points (b) class discriminatory ratio vs. number of PCA components for Z & S dataset using PCA feature extraction technique.



(a) (b)

Figure 4.5: Variance of data points (b) class discriminatory ratio vs number of PCA components for Z & S dataset using ICA feature extraction technique.

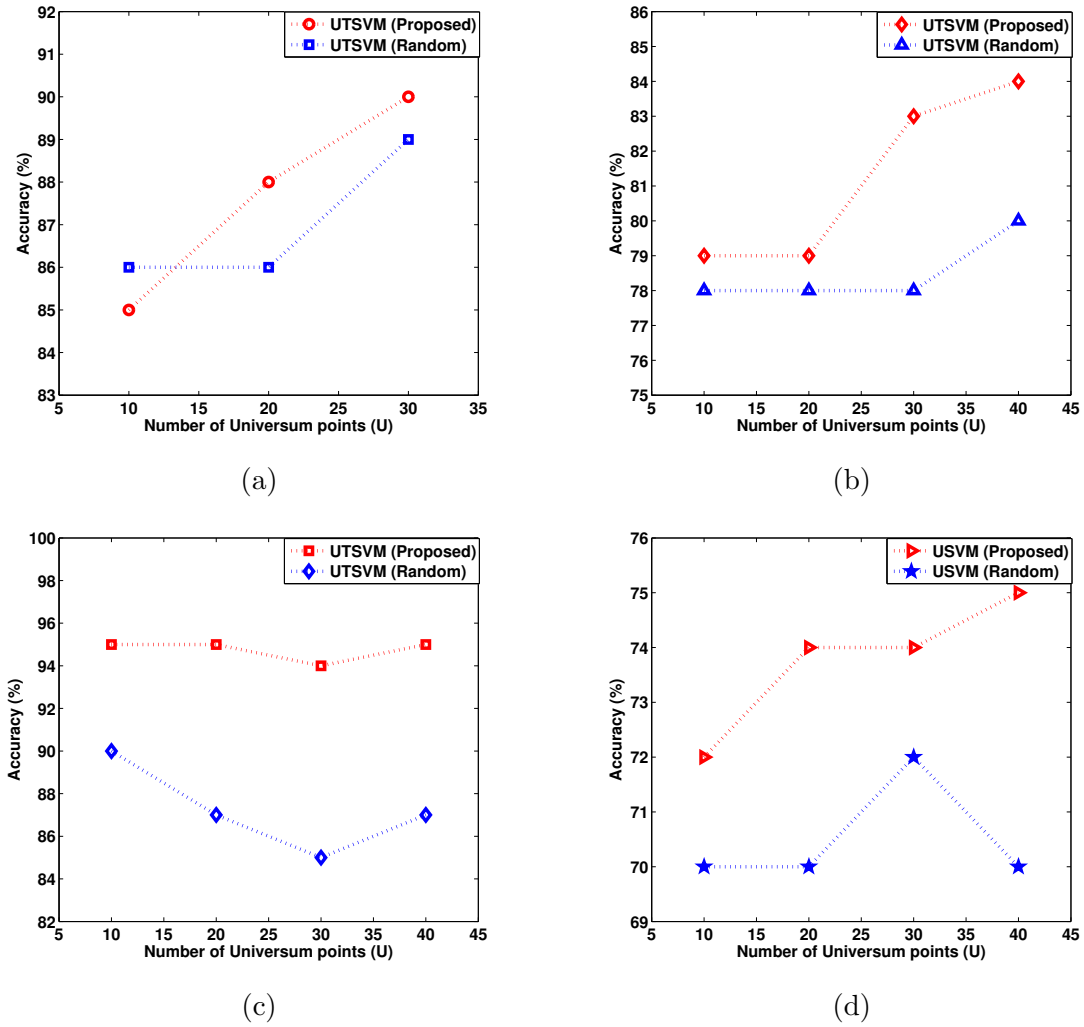


Figure 4.6: Performance comparison of proposed approach for UTSVM with the random averaging method on (a) Z & S using PCA, (b) O & S using PCA, (c) O & S using ICA and (d) O & S using wavelet (db4) feature extraction technique.

The results for all the proposed and baseline methods are shown in terms of prediction accuracy and training time in Table 4.1 and Table 4.2. One can observe from Table 4.1 that the proposed approach outperforms USVM with random averaging, LSSVM and SVM in terms of accuracy. It can be observed in Table 4.1 that LSSVM performs better than SVM and USVM.

From Table 4.2, it is evident that the proposed approach is showing better generalization performance for almost all the datasets as compared to TWSVM and UTSVM. In terms of training time, the proposed approach is comparable with respect to the

Table 4.1: Performance comparison of proposed USVM with SVM, LSSVM and USVM for classification of seizure and healthy EEG signals using RBF kernel.

Dataset (Train size, Test size)	**Feature extraction method**	**SVM** Accuracy(%) (c, μ) Time (s)	**LSSVM** Accuracy(%) (c, μ) Time (s)	**USVM** Accuracy(%) (c, μ, ϵ, u) Time (s)	**Proposed USVM** Accuracy(%) (c, μ, ϵ, u) Time (s)
Z & S $(100 \times 50, 100 \times 50)$	PCA	69 $(10^2, 21180.7)$ 0.09954	73 $(10^5, 21180.7)$ 0.03017	69 $(10^2, 21180.7, 0.2, 15)$ 0.17169	**77** $(10^3, 21180.7, 0.3, 15)$ 0.16816
Z & S $(100 \times 15, 100 \times 15)$	ICA	80 $(10^3, 69.919)$ 0.09963	79 $(10^4, 69.919)$ 0.02733	**81** $(10^3, 69.919, 0.2, 10)$ 0.14142	79 $(10^2, 69.919, 0.1, 10)$ 0.13958
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db4)	69 $(10^2, 21415.9)$ 0.09859	73 $(10^5, 21415.9)$ 0.02984	69 $(10^2, 21415.9, 0.1, 10)$ 0.14816	**76** $(10^4, 21415.9, 0.3, 10)$ 0.14314
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (Haar)	69 $(10^2, 21196.3)$ 0.10009	72 $(10^4, 21196.3)$ 0.0295	69 $(10^3, 21196.3, 0.2, 30)$ 0.25236	**84** $(10^4, 21196.3, 0.7, 30)$ 0.25148
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db2)	69 $(10^2, 21315.9)$ 0.10182	73 $(10^5, 21315.9)$ 0.02925	69 $(10^2, 21315.9, 0.1, 10)$ 0.14347	**76** $(10^4, 21315.9, 0.5, 10)$ 0.14183
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db6)	69 $(10^2, 21503.3)$ 0.09742	71 $(10^2, 21503.3)$ 0.03005	69 $(10^2, 21503.3, 0.1, 10)$ 0.14553	**77** $(10^4, 21503.3, 0.5, 10)$ 0.14177
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db1)	69 $(10^2, 20956.4)$ 0.09981	74 $(10^5, 20956.4)$ 0.031	69 $(10^2, 20956.4, 0.1, 10)$ 0.14285	**78** $(10^4, 20956.4, 0.1, 10)$ 0.14328
O & S $(100 \times 50, 100 \times 50)$	PCA	72 $(10^1, 20400)$ 0.10182	69 $(10^2, 20400)$ 0.02975	67 $(10^3, 20400, 0.1, 40)$ 0.32457	**75** $(10^1, 20400, 0.3, 40)$ 0.30981
O & S $(100 \times 50, 100 \times 50)$	ICA	72 $(10^2, 105.268)$ 0.10174	74 $(10^5, 105.268)$ 0.03955	72 $(10^3, 105.268, 0.3, 20)$ 0.2129	**76** $(10^2, 105.268, 0.6, 20)$ 0.18707
O & S $(100 \times 30, 100 \times 30)$	Wavelet (db4)	71 $(10^1, 20139.2)$ 0.09831	71 $(10^1, 20139.2)$ 0.0284	70 $(10^2, 20139.2, 0.1, 40)$ 0.32568	**75** $(10^1, 20139.2, 0.3, 40)$ 0.31293
O & S $(100 \times 50, 100 \times 50)$	Wavelet (Haar)	70 $(10^2, 19800.4)$ 0.10273	70 $(10^2, 19800.4)$ 0.04194	69 $(10^3, 19800.4, 0.2, 40)$ 0.32389	**75** $(10^1, 19800.4, 0.3, 40)$ 0.31326
O & S $(100 \times 50, 100 \times 50)$	Wavelet (db2)	68 $(10^2, 20074.4)$ 0.09935	69 $(10^2, 20074.4)$ 0.03094	67 $(10^2, 20074.4, 0.1, 40)$ 0.31621	**75** $(10^1, 20074.4, 0.3, 40)$ 0.31399
O & S $(100 \times 50, 100 \times 50)$	Wavelet (db6)	69 $(10^1, 19984.8)$ 0.09922	70 $(10^2, 19984.8)$ 0.02976	69 $(10^2, 19984.8, 0.1, 40)$ 0.31894	**77** $(10^0, 19984.8, 0.1, 40)$ 0.31528
O & S $(100 \times 50, 100 \times 50)$	Wavelet (db1)	71 $(10^1, 20412.5)$ 0.10013	69 $(10^2, 20412.5)$ 0.03019	68 $(10^2, 20412.5, 0.3, 40)$ 0.32286	**76** $(10^0, 20412.5, 0.1, 40)$ 0.31789
Average accuracy		70.5	71.9286	69.7857	**76.8571**
Average rank		3	2.3214	3.5	**1.1786**

Table 4.2: Performance comparison of proposed UTSVM with TWSVM and UTSVM for classification of seizure and healthy EEG signals using RBF kernel.

Dataset (Train size, Test size)	**Feature extraction method**	**TWSVM** Accuracy(%) (c, μ) Time (s)	**UTSVM** Accuracy(%) (c, μ, ϵ, u) Time (s)	**Proposed UTSVM** Accuracy(%) (c, μ, ϵ, u) Time (s)
Z & S $(100 \times 50, 100 \times 50)$	PCA	82 $(10^{-5}, 21180.7)$ 0.0191	89 $(10^{0}, 21180.7, 0.7, 30)$ 0.02529	**90** $(10^{1}, 21180.7, 0.1, 30)$ 0.02599
Z & S $(100 \times 15, 100 \times 15)$	ICA	94 $(10^{0}, 69.919)$ 0.01607	95 $(10^{-2}, 69.919, 0.6, 10)$ 0.01804	**99** $(10^{-5}, 69.919, 0.1, 10)$ 0.01756
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db4)	82 $(10^{-5}, 21415.9)$ 0.01805	78 $(10^{1}, 21415.9, 0.3, 30)$ 0.02533	**91** $(10^{1}, 21415.9, 0.1, 30)$ 0.02509
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (Haar)	79 $(10^{-5}, 21196.3)$ 0.01819	80 $(10^{0}, 21196.3, 0.6, 20)$ 0.02217	**88** $(10^{1}, 21196.3, 0.1, 20)$ 0.02255
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db2)	82 $(10^{-5}, 21315.9)$ 0.01854	89 $(10^{0}, 21315.9, 0.7, 30)$ 0.02468	**90** $(10^{1}, 21315.9, 0.1, 30)$ 0.0251
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db6)	80 $(10^{-5}, 21503.3)$ 0.01813	81 $(10^{0}, 21503.3, 0.7, 20)$ 0.022	**87** $(10^{1}, 21503.3, 0.1, 20)$ 0.02306
Z & S $(100 \times 50, 100 \times 50)$	Wavelet (db1)	80 $(10^{-5}, 20956.4)$ 0.01832	**89** $(10^{0}, 20956.4, 0.7, 30)$ 0.02431	88 $(10^{1}, 20956.4, 0.1, 30)$ 0.0256
O & S $(100 \times 50, 100 \times 50)$	PCA	79 $(10^{-4}, 20400)$ 0.01826	80 $(10^{-4}, 20400, 0.6, 40)$ 0.02571	**84** $(10^{-2}, 20400, 0.6, 40)$ 0.02601
O & S $(100 \times 50, 100 \times 50)$	ICA	94 $(10^{0}, 105.268)$ 0.01699	90 $(10^{-1}, 105.268, 0.6, 10)$ 0.01823	**95** $(10^{-1}, 105.268, 0.1, 10)$ 0.01942
O & S $(100 \times 30, 100 \times 30)$	Wavelet (db4)	**84** $(10^{-3}, 20139.2)$ 0.01822	78 $(10^{0}, 20139.2, 0.2, 20)$ 0.02271	**84** $(10^{-3}, 20139.2, 0.1, 20)$ 0.02236
O & S $(100 \times 50, 100 \times 50)$	Wavelet (Haar)	**82** $(10^{-3}, 19800.4)$ 0.01874	79 $(10^{0}, 19800.4, 0.3, 10)$ 0.02041	**82** $(10^{-3}, 19800.4, 0.1, 10)$ 0.01991
O & S $(100 \times 50, 100 \times 50)$	Wavelet (db2)	**83** $(10^{-3}, 20074.4)$ 0.01829	78 $(10^{-1}, 20074.4, 0.5, 10)$ 0.02005	**83** $(10^{-3}, 20074.4, 0.1, 10)$ 0.02022
O & S $(100 \times 50, 100 \times 50)$	Wavelet (db6)	80 $(10^{-4}, 19984.8)$ 0.02598	77 $(10^{0}, 19984.8, 0.3, 40)$ 0.02778	**85** $(10^{-2}, 19984.8, 0.7, 40)$ 0.02615
O & S $(100 \times 50, 100 \times 50)$	Wavelet (db1)	**84** $(10^{-3}, 20412.5)$ 0.01862	79 $(10^{-1}, 20412.5, 0.5, 10)$ 0.02023	**84** $(10^{-3}, 20412.5, 0.1, 10)$ 0.02013
Average accuracy		83.2143	83	**87.8571**
Average rank		2.3571	2.4286	**1.2143**

existing universum based methods. It is also noticeable from Table 4.1 and 4.2 that the universum based approaches take more computation time as compared to traditional algorithms such as SVM, LSSVM and TWSVM. This additional time is due to the incorporation of universum data points which can be traded for the generalization performance. LSSVM takes very less computation time since it solves a system of linear equations. It is noticeable in Table 4.1 and 4.2 that the existing universum based approaches viz. USVM and UTSVM which use random averaging for universum have not performed better than the other algorithms. This is because the seizure data contains noisy data points, and thus the generated universum data do not reflect the distribution of data. On the other hand, the proposed approach of selecting the universum from interictal EEG signals gives better accuracy in most of the datasets. It is due to the removal of noise in the universum data. This justifies the applicability of the proposed method for classification of seizure and healthy EEG signals.

One can notice from Table 4.1 that the proposed approach has not performed better for all the datasets. So, we analyze the comparative performance of the proposed approach with the existing approaches. The average ranks of SVM, LSSVM, USVM and proposed USVM on the basis of accuracy is also shown in Table 4.1. One can notice that the average rank of the proposed USVM is lowest among all the methods. We perform the Friedman test with the corresponding post-hoc test [172] for the statistical comparison of the performance of the 4 algorithms using 14 datasets. We assume all the methods are equivalent under null hypothesis. By applying formula (2.54), we get the value of $\chi_F^2 = 25.4352$.

The F_F value is calculated as

$$F_F = \frac{(14-1)(25.4352)}{14 \times (4-1) - 25.4352} = 19.9615.$$

where for 4 methods and 14 datasets, the F-distribution has $\big(4-1, (4-1)(14-1)\big) = (3, 39)$ degrees of freedom. For the significance level at $\alpha = 0.05$, the critical value of $F(3, 39)$ is 2.8451. Since $F_F = 19.9615 > 2.8451$, we reject the null hypothesis.

To check the pairwise difference between the proposed and existing algorithms,

we use the Nemenyi posthoc test using Eq. (2.56). For significant pairwise difference between the methods at significance level of $\alpha = 0.10$, the average ranks of the methods shown in Table should differ by atleast $2.291\sqrt{\frac{4(4+1)}{6\times14}} = 1.1179$. The proposed USVM is significantly different from SVM, LSSVM and USVM algorithms.

The accuracy values are shown with the training time for the proposed UTSVM with TWSVM and UTSVM in Table 4.2. One can observe that the proposed UTSVM has shown better generalization performance in most of the cases. Table 4.2 shows the average ranks of TWSVM, UTSVM and proposed UTSVM based on accuracy values. The proposed UTSVM has the lowest rank among all the methods. We further performed the Friedman statistics with the corresponding post-hoc test to find the significant difference between TWSVM, UTSVM and proposed UTSVM.

The Friedman statistic is computed using Table 4.2 under null hypothesis. Similar to proposed USVM, χ_F^2 is calculated as 12.9996. In this case, the F_F value for the F-distribution with $\big(4-1,(3-1)(14-1)\big) = (2,26)$ degrees of freedom is 11.266. For the significance level at $\alpha = 0.05$, the critical value of $F(2,26)$ is 3.3690. Since $F_F = 11.266 > 3.3690$, we reject the null hypothesis. The proposed UTSVM is significantly different from TWSVM and UTSVM. It is noticeable from Table 4.1 and 4.2 that the proposed UTSVM is showing highest generalization performance as compared to the existing methods. The highest accuracy for Z & S is obtained as 99% in the case of ICA feature extraction with the proposed UTSVM. For O & S, the highest accuracy is found with ICA feature extraction technique using the proposed UTSVM.

Fig. 4.7 illustrates the accuracy comparison of different algorithms for the classification of seizure and non-seizure data using different feature extraction techniques. In Fig. 4.8, the insensitivity performance of the proposed approach of USVM is shown for the parameters and C and ϵ. It can be observed that the proposed USVM gives high accuracy for higher values of C and ϵ. The insensitivity performance of the proposed approach with UTSVM is shown in Fig. 4.9. It is evident from Fig. 4.9 that the proposed UTSVM gives better generalization performance for lesser values of C and ϵ.

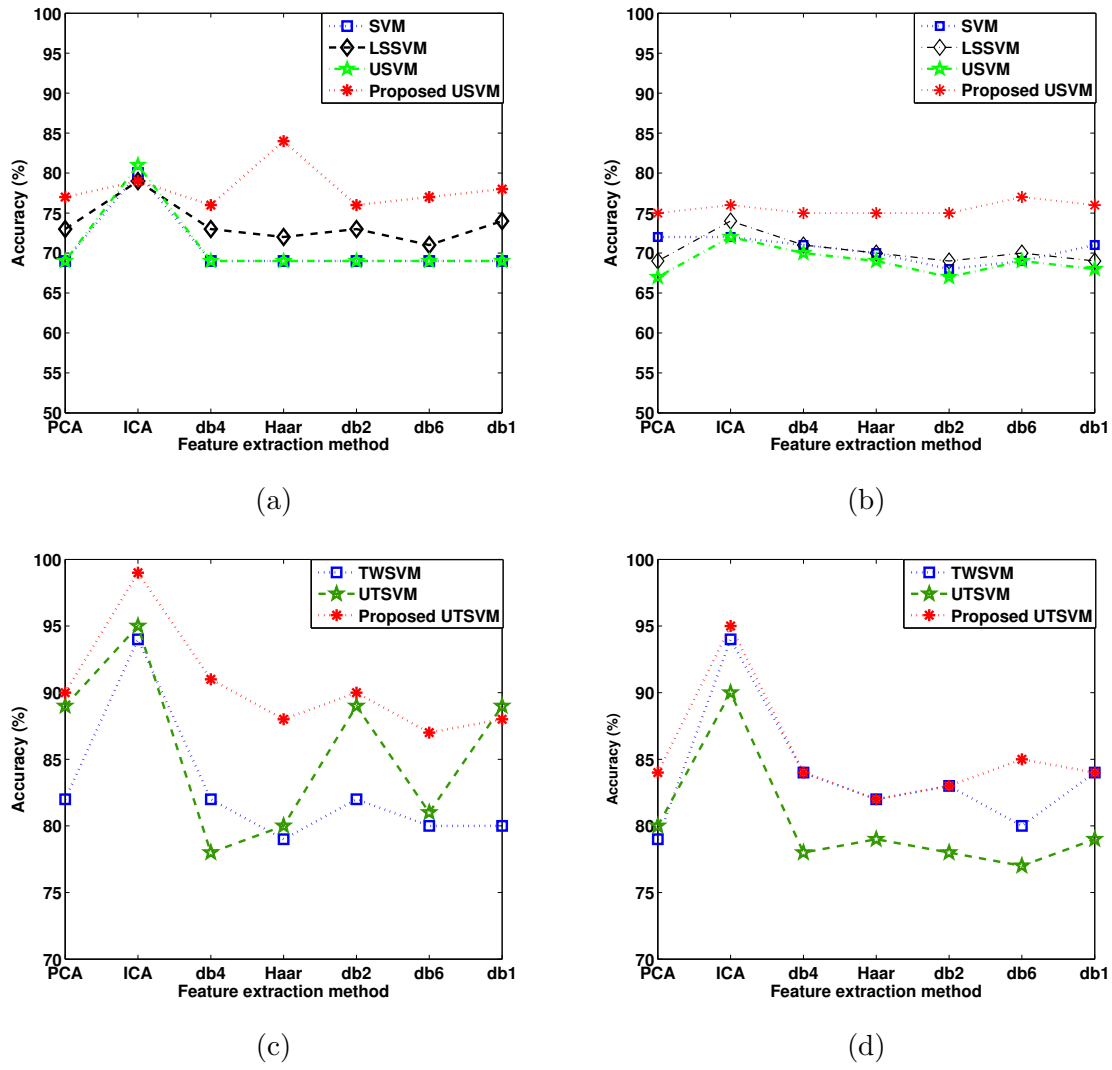
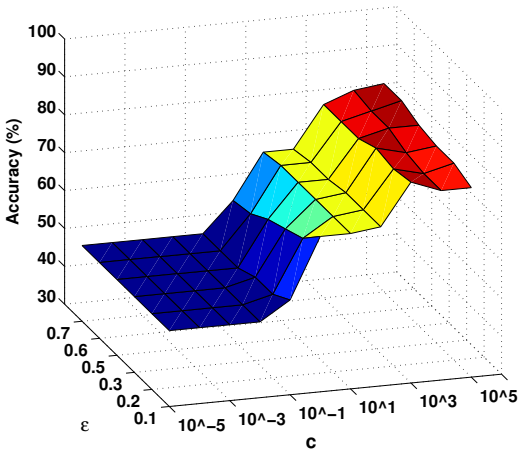
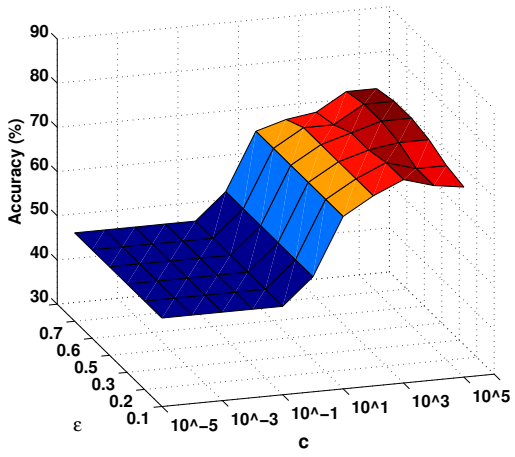


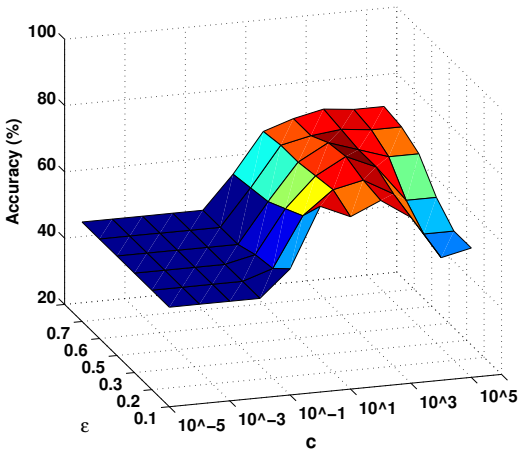
Figure 4.7: Accuracy comparison for classification of EEG signals using different algorithms with RBF kernel. SVM based algorithms for classification on (a) Z & S and (b) O & S datasets, and TWSVM based algorithms on (c) Z & S and (d) O & S datasets using different feature extraction techniques.



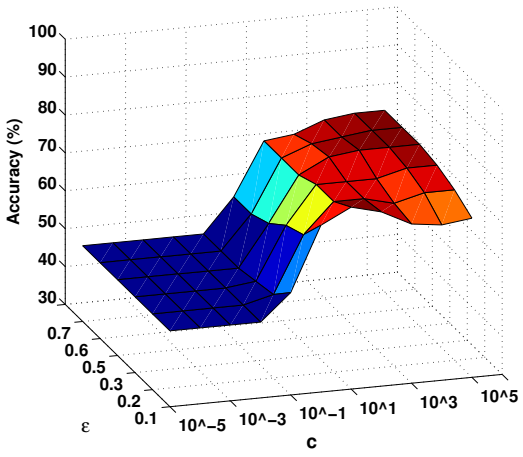
(a) Z & S with Haar



(b) Z & S with db4

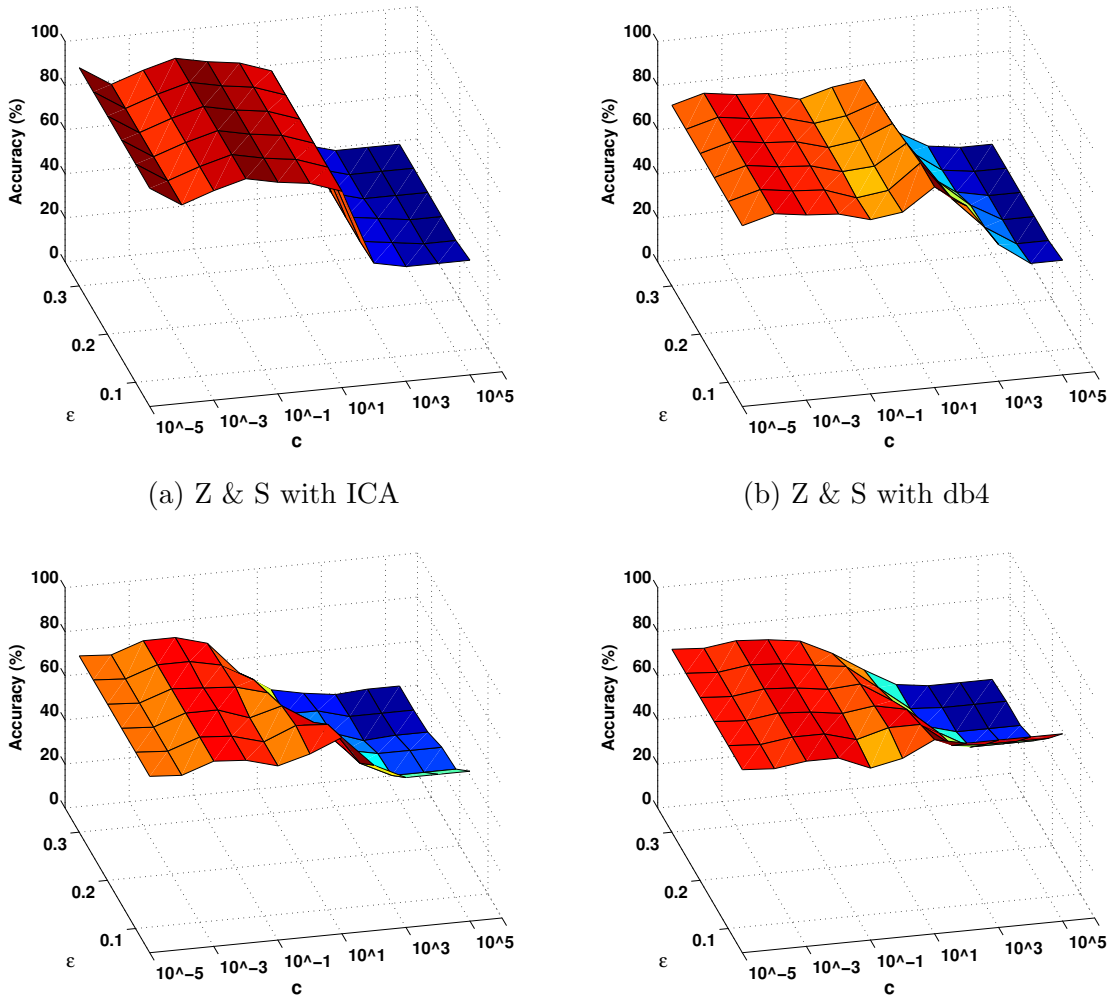


(c) O & S with db1



(d) O & S with db4

Figure 4.8: Insensitivity performance of proposed USVM for classification of seizure and healthy EEG signals to the user specified parameters (c, ϵ) using RBF kernel.



(a) Z & S with ICA

(b) Z & S with db4

(c) O & S with PCA

(d) O & S with db4

Figure 4.9: Insensitivity performance of proposed UTSVM for classification of seizure and healthy EEG signals to the user specified parameters (c, ϵ) using RBF kernel.

In the following section, we present a feature selection algorithm based on universum support vector machine with application on another brain disorder i.e., Alzheimer's disease.

4.2 Diagnosis of Alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE)

Alzheimer's disease is one of the most common causes of death in today's world. Magnetic resonance imaging (MRI) provides an efficient and non-invasive approach for diagnosis of Alzheimer's disease. Efficient feature extraction techniques are needed for accurate classification of MRI images. Motivated by the work on support vector machine based recursive feature elimination (SVM-RFE) (Guyon et al., 2002 [1]), we propose a novel feature selection technique to incorporate prior information about data distribution in the recursive feature elimination process. In the subsequent sections, first we describe the data and the feature extraction methods, followed by formulation of the proposed algorithm with analysis of the results.

4.2.1 Data

All data used in this work were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). ADNI was launched in the year 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The main objective of ADNI is to analyze the effectiveness of neuroimaging techniques like magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical neuropsychological tests to estimate the onset of Alzheimer's disease from the state of mild cognitive impairment. For more information, visit `www.adni-info.org`.

4.2.2 Image acquisition

A total of 150 T1-weighted structural MRI (sMRI) images are downloaded from ADNI database. Each of the categories i.e., CN, MCI, and AD comprises of 50 subjects. The subjects are within the age range of 60-90 with mean age of 75.83, and standard deviation of 6.07. The Mini-Mental State Examination (MMSE) score of subjects is in the range of 17-30 with mean and standard deviation of 26.51±2.88. The detailed demographics of cohort are given in Table 4.3.

Images of the following specifications are acquired from the ADNI archive: field strength=1.5 T; description=MP-RAGE; acquisition=3D; pulse sequence= RM; slice thickness=1.2 mm; flip angle=8 degrees; acquisition plane=sagittal, manufacturer=GE medical systems.

Diagnosis	**Age**	**Gender**	**MMSE**
CN	76.65 ± 4.30	39M/11F	29.02 ± 1.15
MCI	75.23 ± 7.02	26M/24F	26.9 ± 1.96
AD	75.60 ± 6.58	28M/22F	23.62 ± 2.24

Table 4.3: Subject demographics.

We also downloaded 817 sMRI images from the ADNI baseline dataset [196, 197] to verify the applicability of the proposed method.

4.2.3 Voxel based morphometry (VBM)

A frequently used neuroimaging toolbox i.e., Statistical Parametric Mapping (SPM) version 12 (Wellcome Trust Centre for Neuroimaging, University College London, U.K.) is used to perform the VBM analysis. The preprocessed data is used for three different binary classification tasks i.e., CN vs AD, CN vs MCI, and MCI vs AD. The scans from each subject category were randomly divided into training and testing sets of 40 and 10 images respectively.

All raw images are aligned in the same coordinate space by setting the origin of the raw scans manually to the anterior commissure (AC), and registering with SPM's single subject T1 template. The registered images are processed by SPM's unified

segmentation routine to segment the images into GM, WM and CSF, and create the template using DARTEL [198] approach. This template is used to normalize the images into the Montreal Neurological Institute (MNI) space with modulation. A Gaussian kernel with full width at half maximum (FWHM) of 8 mm is used for smoothing. All the images are transformed to a dimension of $121 \times 145 \times 121$ with a voxel size of 1.5 mm^3.

A two sample T-test is used for finding the statistically significant voxels by keeping subject age and gender as covariates in the general linear model (GLM) [199]. The differences of individual head sizes are controlled by introducing total intracranial volume (TIV) as a covariate of no interest [200] using the analysis of covariance (ANCOVA)-by-subject approach. The T-test analysis is done by using a p-value of 0.05 with family-wise error (FWE) correction, and an extent threshold of 0 adjacent voxels. The complete approach is shown in Fig. 4.10, and the specified GLM is given in Fig. 4.11. The voxels of interest (VOI) retrieved after statistical analysis of training images are used as masks for specifying voxel coordinates that are significantly different between subject groups [98]. Figs. 4.12 and 4.13 show significant voxels of CN vs AD, and CN vs MCI analysis respectively.

In some previous works [201, 202], all the acquired MRI images were used for creating DARTEL template. In this work, we use DARTEL pipeline for training and testing phase separately. This procedure is followed in all the cases i.e., CN vs AD, CN vs MCI, and MCI vs AD with different features i.e., GM, WM, and CSF. As a whole, the DARTEL approach is used on 18 sets of data (9 training, 9 testing), leading to distinct subject specific templates for training and testing. This is as per real world scenarios where testing images are not available beforehand. The mask from the training set is applied on the testing images for feature extraction [98].

After extracting significant voxel features from the masked images using SPM, we applied PCA [98] and F-score [203] technique for dimensionality reduction and feature selection respectively. The F-score is calculated for all the features using ratio of the variance between the classes and within the classes.

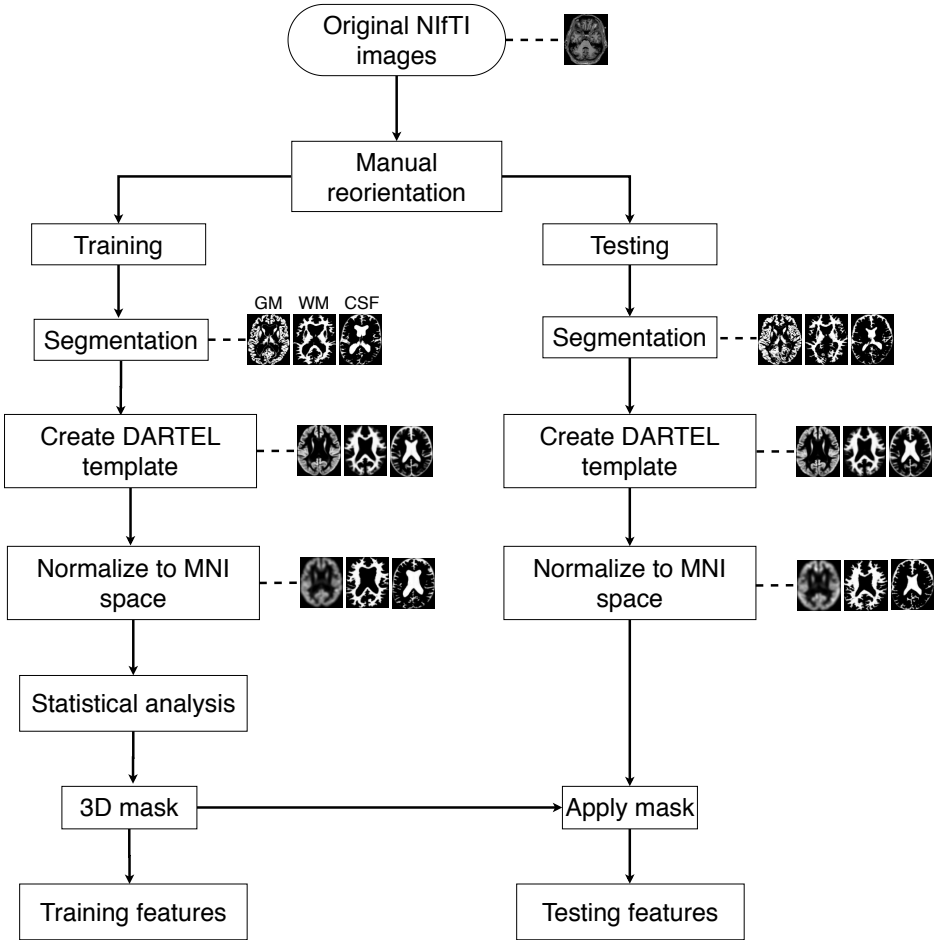


Figure 4.10: VBM image preprocessing pipeline.

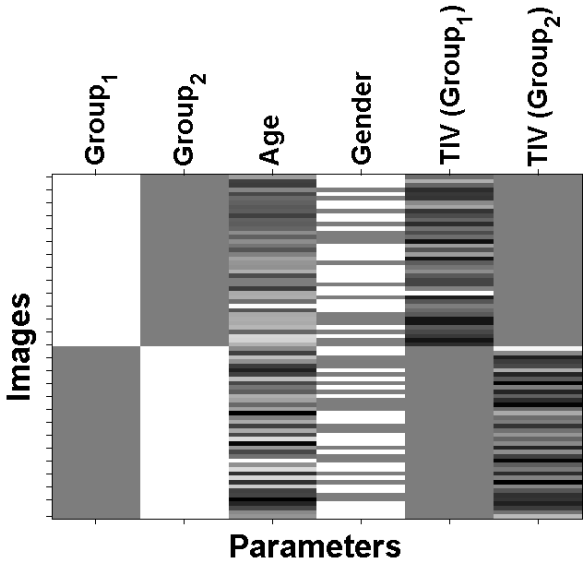
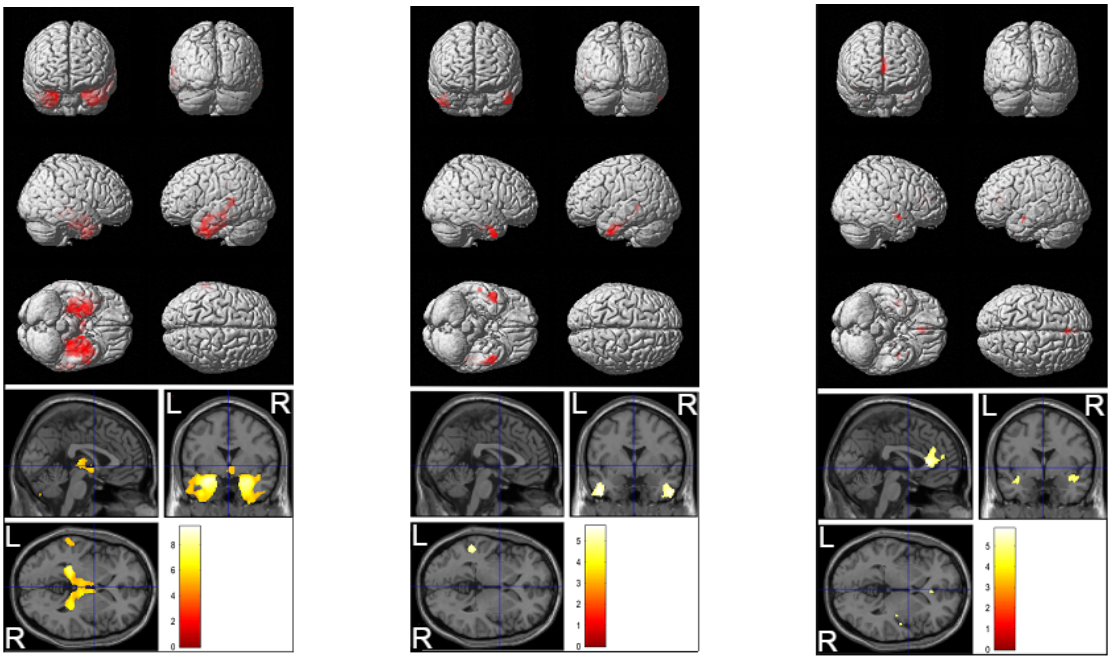
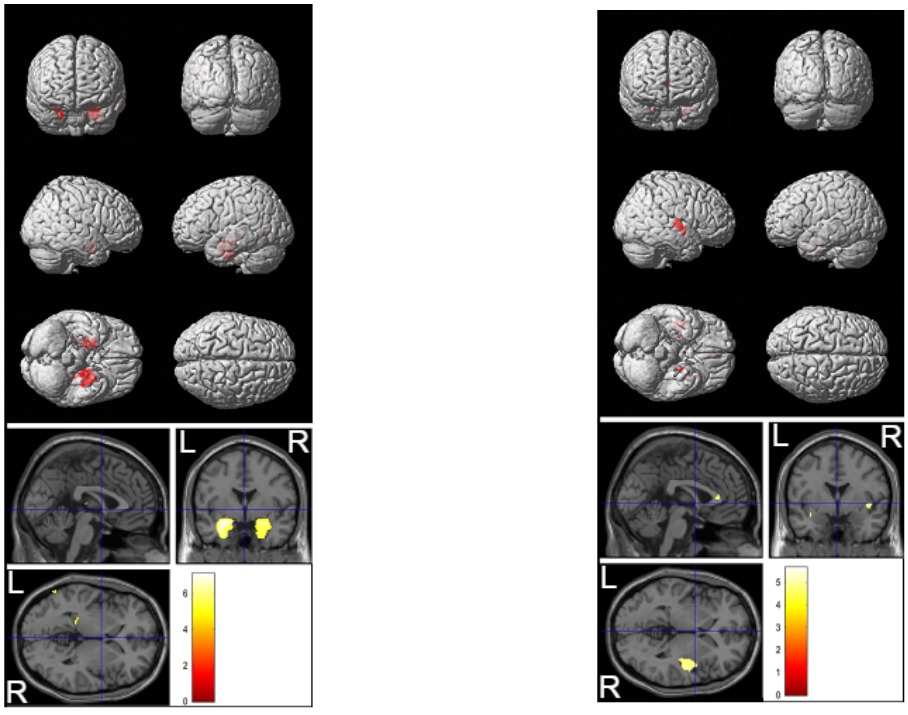


Figure 4.11: GLM design matrix for statistical analysis of gray matter in CN (Group_1) vs AD (Group_2).



(a) GM (b) WM (c) CSF

Figure 4.12: Plot showing significant voxels obtained from VBM analysis for CN vs AD. 3D illustrations (top) are shown for VOI in the MRI images (bottom).



(a) GM (b) CSF

Figure 4.13: Plot showing significant voxels obtained from VBM analysis for CN vs MCI. 3D illustrations (top) are shown for VOI in the MRI images (bottom).

4.2.4 Volume based morphometry (VolBM)

For VolBM analysis, Freesurfer's recon-all pipeline (version 6.0.1) [204, 205] is used on structural MRI images. Out of 150 MRI images, 1 MCI image failed to process in Freesurfer. So, feature selection was performed on 149 images. We extracted 23 subcortical tissue volumes (SCV), 34 WM tissue volumes (WMV), and 34 cortical thickness (CT) measures of every subject. To check the performance of the proposed model on an independent dataset, we downloaded 817 sMRI images from ADNI baseline dataset [196, 197], out of which 4 images failed to process through Freesurfer pipeline. Thus, our baseline dataset includes 228 CN, 398 MCI, and 187 AD images.

Thickness measures from both the brain hemispheres are added together to form the cortical thickness features. A similar approach is used for volumetric features. The volumetric features are normalized by dividing by TIV of the subjects [204, 206]. The variations in neurological features of the subjects are illustrated in Fig. 4.14, and the complete list of features obtained from Freesurfer is given in Table 4.4.

4.2.5 Proposed universum support vector machine based recursive feature elimination (USVM-RFE)

The SVM-RFE algorithm lacks the knowledge about the distribution of data. Moreover, in the recursive process, the SVM classifier eliminates features on the basis of weights for the maximal margin. In order to incorporate knowledge about the data distribution, we use universum samples as shown in Fig. 1.2. The resulting universum based algorithm uses this prior information about distribution of data in the recursive elimination of features leading to better feature selection.

4.2.5.1 Universum data

The universum data is used to align the classifier with the data distribution. As shown in Fig. 1.2, the classifier generated by USVM is better aligned to classify the data points. This helps in the classification of testing data. Without this knowledge of the data distribution, the SVM classifier only tries to maximize the margin. This

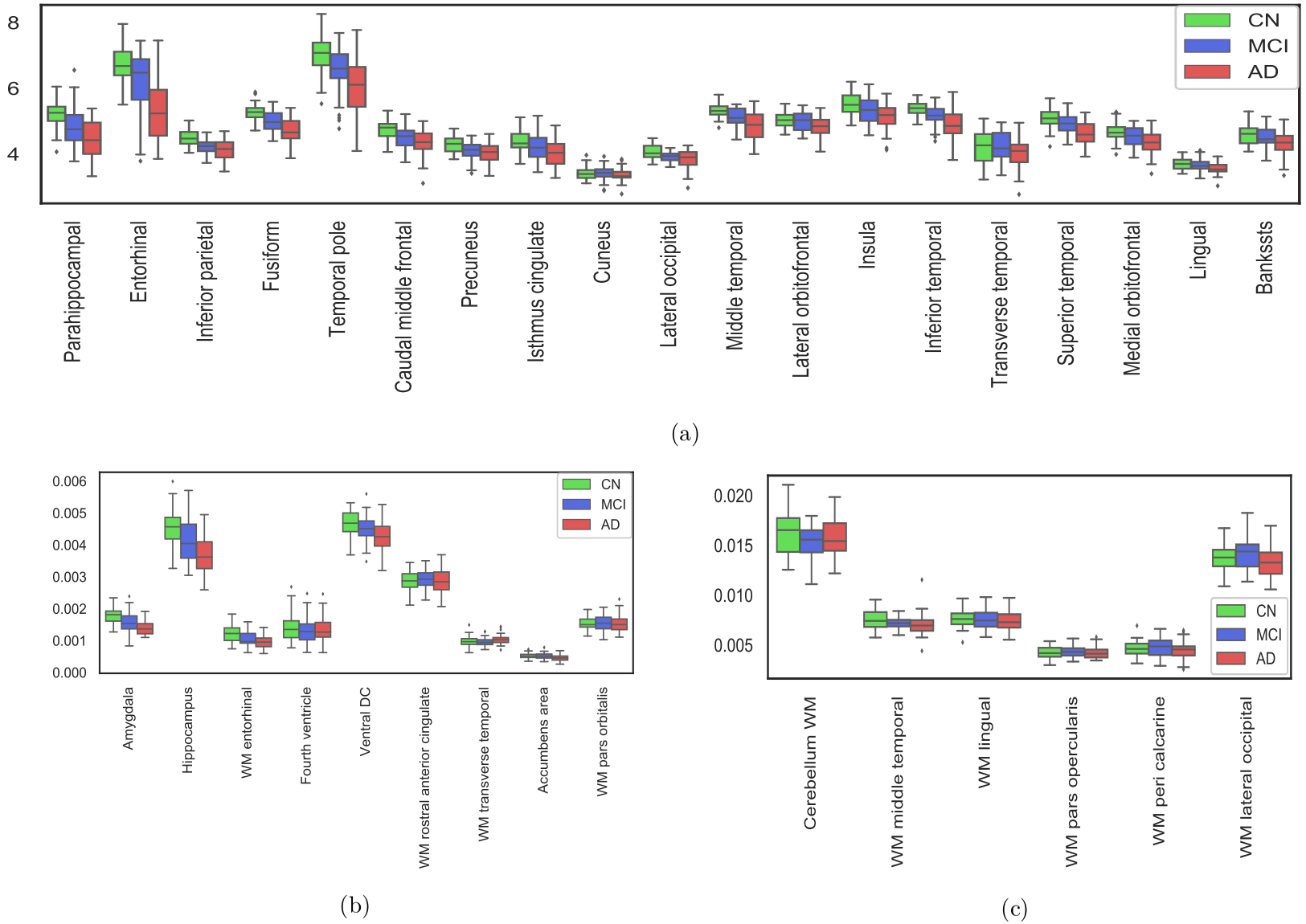


Figure 4.14: Box plot showing distribution of (a) CT, (b) SCV, and (c) WMV obtained from VolBM in the subjects.

Table 4.4: Cortical, subcortical, and white matter features with their feature IDs obtained from VolBM analysis.

Feature ID	Cortical thickness	Feature ID	Subcortical volume	Feature ID	White matter volume
1	Bankssts*	35	Third ventricle	58	WM caudal anterior cingulate*
2	Inferior temporal	36	Fourth ventricle	59	WM caudal middle frontal
3	Middle temporal	37	Brain stem	60	WM Bankssts
4	Superior temporal	38	Inferior lateral ventricle	61	WM inferior temporal
5	Temporal pole	39	Lateral ventricle	62	WM inferior parietal
6	Transverse temporal	40	Ventral DC*	63	WM middle temporal
7	Caudal anterior cingulate	41	CSF	64	WM superior temporal
8	Caudal middle frontal	42	CC posterior*	65	WM superior parietal
9	Cuneus	43	CC mid posterior	66	WM superior frontal
10	Precuneus	44	CC central	67	WM temporal pole
11	Entorhinal	45	CC mid anterior	68	WM transverse temporal
12	Fusiform	46	CC anterior	69	WM cuneus
13	Inferior parietal	47	Accumbens area	70	WM precuneus
14	Superior parietal	48	Amygdala	71	WM entorhinal
15	Isthmus cingulate	49	Caudate	72	WM fusiform
16	Lateral occipital	50	Cerebellum white matter	73	WM isthmus cingulate
17	Lateral orbitofrontal	51	Cerebellum cortex	74	WM lateral occipital
18	Medial orbitofrontal	52	Cerebral white matter	75	WM lateral orbitofrontal
19	Lingual	53	Hippocampus	76	WM medial orbitofrontal
20	Parahippocampal	54	Putamen	77	WM lingual
21	Paracentral	55	Pallidum	78	WM parahippocampal
22	Pars opercularis	56	Thalamus proper	79	WM paracentral
23	Pars triangularis	57	Cortex	80	WM postcentral
24	Peri calcarine			81	WM precentral
25	Post central			82	WM pars opercularis
26	Posterior cingulate			83	WM pars triangularis
27	Precentral			84	WM pericalcarine
28	Pars orbitalis			85	WM pars orbitalis
29	Rostral middle frontal			86	WM posterior cingulate
30	Rostral anterior cingulate			87	WM rostral middle frontal
31	Superior frontal			88	WM rostral anterior cingulate
32	Supramarginal			89	WM supra marginal
33	Frontal pole			90	WM frontal pole
34	Insula			91	WM insula thickness

* Bankssts: Banks of superior temporal sulcus, CC: Corpus callosum, WM: White matter, DC: Diencephalon.

results in reduced generalization performance of the model. In this work, we generated universum samples using random averaging of data points [20, 27].

4.2.5.2 Iterative procedure

In the proposed USVM-RFE, we use the universum data points in each iteration. This results in selection of important features due to addition of universum constraints. The universum data points are constrained to lie within an ϵ-insensitive tube as shown in Fig. 1.2. Since USVM-RFE is a wrapper [207] method, the process of USVM-RFE is divided into three phases: parameter selection, feature elimination, and classification. The proposed universum based USVM-RFE algorithm is described in Alg. 4.1.

We used k-fold cross validation for selecting optimal parameters for feature elimination. Then, the features are eliminated in an iterative manner. Lastly, the classifi-

Algorithm 4.1 Proposed USVM-RFE

Inputs:

Training data

$X = [x_1, x_2, ..., x_l]^T$

Class labels

$Y = [y_1, y_2, ..., y_l]^T$

Universum data

$U = [u_1, u_2, ..., u_r]^T$

Process:

Find optimal parameters for recursive process using k-fold cross validation

Feature set

$S = [1, 2, ..., n]$

Feature ranked list

$R = [\ ]$

REPEAT UNTIL $S = [\ ]$

Feature selection

$X = X(:, S)$

$U = U(:, S)$

Train SVM classifier using parameters obtained in step 11

$\alpha = USVM_{train}(X, Y, U)$

Compute the weight vector with dimension of length(S)

$w = \sum_{i=1}^{l+2r} \alpha_i y_i x_i$

Compute ranking criteria

$C_i = (w_i)^2,\ i = 1, 2, ..., length(S)$

Find feature with smallest rank

$f = argmin(C)$

Update feature ranked list

$R = [S(f), R]$

Eliminate the feature with smallest rank

$S = [1 : f - 1, f + 1 : length(S)]$

Output:

Feature ranked list R.

cation is performed on each of the feature subsets. For computational efficiency, more than one feature can be eliminated in one iteration in the proposed USVM-RFE. The proposed USVM-RFE provides global information about data distribution as compared to the greedy approach in SVM-RFE. However, proper selection of universum is a topic of research [10, 20].

The proposed universum based feature selection is useful for applications such as classification of MRI images, where only few voxels corresponding to specific regions are helpful in the classification. So, we present the application of USVM-RFE on classification of Alzheimer's disease. For identifying brain regions with neurodegeneration, feature selection is performed on VBM as well as VolBM features. USVM is trained on the feature sets obtained after each iteration using k-fold cross validation, and tested on testing data. The feature set with highest accuracy is selected as optimal.

4.2.6 Experimental results

The experiments are carried out on feature sets obtained from VBM as well as VolBM analysis. For VBM, SPM version 12 is used, while for VolBM, Freesurfer version 6.0.1 is used to process the images. The softwares used in generation of 2D and 3D brain overlays are: ITK Snap (v3.8.0-beta) [208], Paraview (v5.6.0) [209], Mricron (`www.nitrc.org/projects/mricron`) and Mricrogl (v1.0.20180623) (`www.nitrc.org/projects/mricrogl`). We used WFU_PickAtlas (v3.0.5b) for selecting ROIs from AAL atlas [210].

The image processing is carried out on a workstation with Windows 10 OS, 64-bit, running on 2.30 GHz Intel ® Xeon processor, and 128 GB RAM. The optimal parameters for the recursive process are obtained using 5-fold cross validation in all datasets. Linear kernel is used in both SVM and USVM for feature extraction as well as classification. The number of PCA components are chosen so as to account for 99% of variance in the data. The number of F-score features is selected as 500. In all cases, the features in training and testing data are normalized to a mean of 0 and standard deviation of 1 using Z-score [206]. The parameter selection is performed based on the following settings.

4.2.6.1 Parameter settings

In some previous works, fixed value of c is chosen for feature selection [1, 117]. However, in this work we used different types of features with varying dimensions. So, in all the VBM features viz., GM, WM, and CSF, grid search is used for obtaining the optimal parameters. The values of penalty parameters $c = c_u$ are selected from the range $\{10^{-6}, 10^{-5}, ..., 10^{0}\}$ for SVM and USVM. This range is selected to avoid overfitting of classifier [211] for high dimensional datasets. For USVM, number of universum samples i.e., u is selected from the set $\{0.1, 0.2\}$, and ϵ is selected from $\{0.3, 0.5, 0.7\}$.

In the RFE phase of VBM, the features are reduced by percentage (per) of feature size for computational efficiency [1] using the following criteria:

$REPEAT\ UNTIL\ (No\_of\_features \geq ceil\ (0.01 * Total\_feature))$

{

$No\_of\_features = per * No\_of\_features$

$if\ (No\_of\_features < ceil\ (0.1 * Total\_feature))$

$per = 0.995;$

$else\ if(No\_of\_features < ceil\ (0.5 * Total\_feature))$

$per = 0.99;$

$else\ (No\_of\_features < ceil\ (0.7 * Total\_feature))$

$per = 0.98;$

}

where *ceil* is the ceiling function.

In all VolBM feature sets i.e., CT, SCV, and WMV, $c = c_u$ is chosen from $\{10^{-6}, 10^{-5}, ..., 10^{4}\}$ in both feature selection and classification phase. For USVM, for parameter selection, u is selected from the set $\{0.1, 0.3, 0.45\}$, and ϵ is selected from $\{0.6, 0.7, 0.8\}$. In the classification phase, for features of different dimensions u is selected from the set $\{0.1, 0.15, 0.35, 0.45\}$, and ϵ is selected from $\{0.3, 0.5, 0.6, 0.8\}$. In the RFE process, the features are reduced one at a time, due to less size of VolBM feature set.

To check the performance of proposed USVM-RFE on other applications, we performed experiments on two UCI [170] biomedical datasets i.e., Wpbc (Breast Cancer Wisconsin Prognostic), and Wdbc (Breast Cancer Wisconsin Diagnostic). The parameters c, c_u are selected by varying values from the set $\{10^{-5}, 10^{-4}, ..., 10^{5}\}$ for SVM and USVM. For USVM, the values for u and ϵ are set as 0.3 and 0.5 respectively. Experimental results on classification of different subject groups i.e., CN, MCI, and AD are shown for different features sets in the following subsections.

4.2.6.2 VBM features

We performed experiments on feature selection from VBM features obtained from SPM toolbox for both SVM-RFE and proposed USVM-RFE.

I. CN vs AD:

The comparison of classification accuracy for CN vs AD is presented in Table 4.5 for GM, WM, and CSF features sets. It is observable that for GM features, the proposed USVM-RFE outperforms SVM-RFE w.r.t. accuracy and sensitivity in all the 7 reduced feature sets. This is a result of prior knowledge in USVM-RFE about the data distribution. However, the accuracy of USVM-RFE reached a maximum of 90% for 20% features, and then declined in lower dimensional features due to loss of informative voxels. For WM, both methods have similar performance. In case of CSF, USVM-RFE performed better for lower dimensional feature set (1% features), while SVM-RFE is having high accuracy for high dimensional feature set (50% features).

The classification accuracy of SVM and USVM for full feature sets of VBM is shown in Table 4.6. The optimal parameters for RFE process are also shown. These parameters are utilized to perform the feature elimination process. Moreover, Table 4.6 also shows the accuracies obtained after feature reduction by PCA, and F-score. From Tables 4.5 and 4.6, one can observe that the proposed USVM-RFE performs better than PCA and F-score in the feature selection.

II. CN vs MCI:

For CN vs MCI, the results for feature selection are shown in Table 4.7. It is

Table 4.5: Performance comparison of proposed USVM-RFE with SVM-RFE based on classification accuracy (%) for CN vs AD on reduced VBM feature sets. Bold values indicate highest accuracy for the dataset, and underlined values show highest accuracy of the algorithm.

Features (%)	Number of features	Proposed USVM-RFE			SVM-RFE		
		Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Gray matter							
1	264	**80**	80	80	70	70	70
5	1323	**75**	90	60	70	70	70
10	2628	**85**	90	80	70	70	70
15	3952	**85**	90	80	75	60	90
20	5304	**90**	90	90	75	70	80
30	7953	**90**	90	90	80	80	80
50	13045	**90**	90	90	75	70	80
White matter							
1	26	**70**	60	80	60	40	80
5	133	70	60	80	**75**	80	70
10	265	**75**	80	70	**75**	80	70
15	398	**70**	70	70	**70**	70	70
20	534	**70**	70	70	**70**	70	70
30	794	70	80	60	**75**	80	70
50	1333	**70**	70	70	**70**	80	60
CSF							
1	18	**70**	50	90	65	40	90
5	90	**65**	40	90	**65**	40	90
10	180	**65**	40	90	**65**	40	90
15	269	**65**	40	90	**65**	40	90
20	359	**65**	40	90	**65**	40	90
30	535	**60**	40	80	**60**	40	80
50	895	65	50	80	**75**	80	70

Table 4.6: Performance comparison of USVM with SVM is shown based on classification accuracy (%) for CN vs AD on VBM features. The optimal parameters are shown in parentheses.

Features	Number of features	USVM			SVM		
		Accuracy ($c = c_u, \epsilon, u$)	Sensitivity	Specificity	Accuracy (c)	Sensitivity	Specificity
Gray matter							
All features	26524	**85** $(10^{-3}, 0.3, 0.2)$	90	80	70 (10^{-6})	80	60
PCA	59	75 $(10^{0}, 0.3, 0.2)$	80	70	**80** (10^{-1})	90	70
F-score	500	**70** $(10^{-4}, 0.5, 0.2)$	70	70	**70** (10^{-4})	70	70
White matter							
All features	2675	**75** $(10^{-5}, 0.3, 0.2)$	90	60	**75** (10^{-5})	90	60
PCA	26	60 $(10^{-2}, 0.3, 0.2)$	40	80	**65** (10^{-1})	50	80
F-score	500	**70** $(10^{-4}, 0.3, 0.2)$	70	70	**70** (10^{-4})	70	70
CSF							
All features	1802	70 $(10^{-4}, 0.3, 0.1)$	60	80	**75** (10^{-4})	70	80
PCA	23	**60** $(10^{0}, 0.5, 0.1)$	30	90	**60** (10^{2})	30	90
F-score	500	**60** $(10^{-4}, 0.5, 0.1)$	40	80	**60** (10^{-4})	40	80

Table 4.7: Performance comparison of proposed USVM-RFE with SVM-RFE is shown based on classification accuracy (%) for CN vs MCI on reduced feature sets of VBM.

Features (%)	Number of features	Proposed USVM-RFE			SVM-RFE		
		Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Gray matter							
1	64	**70**	80	60	65	70	60
5	322	**75**	90	60	70	80	60
10	647	**70**	80	60	**70**	80	60
15	969	70	80	60	**75**	90	60
20	1285	**70**	80	60	**70**	80	60
30	1925	**70**	80	60	**70**	80	60
50	3179	**70**	80	60	**70**	80	60
White matter							
1	26	**50**	20	80	45	10	80
5	133	**60**	60	60	**60**	60	60
10	265	**65**	60	70	**65**	60	70
15	398	55	30	80	**65**	60	70
20	534	40	10	70	**55**	40	70
30	794	**50**	20	80	45	20	70
50	1333	**55**	30	80	45	10	80
CSF							
1	14	**90**	100	80	80	100	60
5	70	**85**	100	70	**85**	100	70
10	140	**85**	100	70	80	100	60
15	212	80	100	60	**85**	100	70
20	281	80	100	60	**85**	100	70
30	424	**85**	90	80	**85**	100	70
50	700	**85**	100	70	80	100	60

visible that in case of GM, USVM-RFE gives accuracy of 75% with sensitivity of 90% for lesser features i.e., 5%, as compared to SVM-RFE (15% features). However, for CSF voxels, highest accuracy of 90% is obtained by proposed USVM-RFE for 1% features, whereas SVM-RFE obtains lower accuracy (85%) with a higher feature size (5%). In case of WM also, USVM-RFE has outperformed SVM-RFE in most cases. Table 4.8 shows the classification results on full feature sets. One may notice that USVM has performed better than SVM for full features on CSF features. SVM and USVM have also shown lesser accuracies than PCA and F-score in some cases. This may be attributed to overfitting of SVM and USVM for high dimensional feature sets [211]. However, both algorithms have shown improvement on accuracy in the RFE process.

III. MCI vs AD:

Table 4.9 shows the classification accuracies for MCI vs AD on reduced VBM feature sets. One can notice that USVM-RFE performs better than SVM-RFE for lower dimensional features of GM. In case of WM, SVM-RFE performed better than

Table 4.8: Performance comparison of USVM with SVM is shown based on classification accuracy (%) for CN vs MCI on VBM features. The optimal parameters are shown in parentheses.

Features	Number of features	USVM Accuracy ($c = c_u, \epsilon, u$)	USVM Sensitivity	USVM Specificity	SVM Accuracy (c)	SVM Sensitivity	SVM Specificity
Gray matter							
All features	6478	**70** ($10^{-5}, 0.3, 0.1$)	80	60	**70** (10^{-4})	80	60
PCA	37	65 ($10^{1}, 0.3, 0.2$)	50	80	**70** (10^{1})	50	90
F-score	500	**75** ($10^{-4}, 0.7, 0.1$)	90	60	70 (10^{-3})	80	60
White matter							
All features	2675	**45** ($10^{-3}, 0.7, 0.1$)	10	80	**45** (10^{-3})	10	80
PCA	28	**55** ($10^{1}, 0.3, 0.1$)	10	100	**55** (10^{1})	10	100
F-score	500	**60** ($10^{-3}, 0.3, 0.1$)	60	60	**60** (10^{-3})	60	60
CSF							
All features	1416	**85** ($10^{-4}, 0.3, 0.2$)	100	70	80 (10^{-4})	90	70
PCA	22	60 ($10^{1}, 0.7, 0.1$)	70	50	**65** (10^{1})	70	60
F-score	500	75 ($10^{-4}, 0.3, 0.1$)	80	70	**85** (10^{-3})	100	70

USVM-RFE. This may be attributed to improper universum data generated using random averaging scheme. Since the MCI vs AD dataset is non-linear in nature, random averaging may generate improper universum data. In case of CSF, both SVM-RFE and USVM-RFE have shown low classification accuracy. However, USVM-RFE has performed better in comparison to SVM-RFE. Moreover, Table 4.10 shows the performance on the full feature set. It can be seen that USVM performs better than SVM in most cases. Also, in Table 4.9 the classification accuracy of MCI vs AD is low in both SVM-RFE and USVM-RFE. This is attributed to the relatively difficult classification problem of MCI vs AD in comparison to CN vs AD, and CN vs MCI. [98]. The discriminating regions in CN vs AD are more as compared to MCI vs AD. This is due to similar distribution of data points in MCI and AD subjects [125].

Remark: No significant voxels were found in case of WM features of CN vs MCI, and all cases of MCI vs AD. Thus, the masks obtained from CN vs AD experiments were used to select region of interests (ROI) in these cases [98].

Table 4.9: Performance comparison of proposed USVM-RFE with SVM-RFE is shown based on classification accuracy (%) for MCI vs AD on reduced feature sets of VBM.

Features (%)	Number of features	Proposed USVM-RFE			SVM-RFE		
		Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Gray matter							
1	264	**60**	50	70	55	40	70
5	1323	**65**	60	70	55	40	70
10	2628	**65**	60	70	60	50	70
15	3952	**60**	50	70	**60**	50	70
20	5304	**60**	50	70	**60**	50	70
30	7953	55	50	60	**60**	50	70
50	13045	**60**	50	70	**60**	50	70
White matter							
1	26	45	50	40	**50**	60	40
5	133	**55**	70	40	50	60	40
10	265	**50**	100	0	**50**	60	40
15	398	50	100	0	**65**	100	30
20	534	50	100	0	**65**	90	40
30	794	50	90	10	**60**	100	20
50	1333	55	90	20	**65**	90	40
CSF							
1	18	**45**	0	90	35	40	30
5	90	**40**	0	80	**40**	40	40
10	180	**45**	0	90	40	50	30
15	269	**45**	0	90	**45**	60	30
20	359	40	0	80	**45**	70	20
30	535	**50**	40	60	45	70	20
50	895	**55**	50	60	45	50	40

Table 4.10: Performance comparison of USVM with SVM is shown based on classification accuracy (%) for MCI vs AD on VBM features. The optimal parameters are shown in parentheses.

Features	Number of features	USVM			SVM		
		Accuracy ($c = c_u, \epsilon, u$)	Sensitivity	Specificity	Accuracy (c)	Sensitivity	Specificity
Gray matter							
All features	26524	**70** $(10^{-6}, 0.7, 0.2)$	60	80	60 (10^{-5})	60	60
PCA	60	**50** $(10^{-6}, 0.5, 0.1)$	30	70	**50** (10^{-2})	30	70
F-score	500	**65** $(10^{-4}, 0.3, 0.1)$	60	70	60 (10^{-3})	50	70
White matter							
All features	2675	55 $(10^{-5}, 0.3, 0.2)$	70	40	**65** (10^{-3})	90	40
PCA	26	**40** $(10^{0}, 0.5, 0.2)$	50	30	40 (10^{-1})	50	30
F-score	500	40 $(10^{2}, 0.3, 0.1)$	80	0	**50** (10^{0})	100	0
CSF							
All features	1802	**45** $(10^{-1}, 0.3, 0.2)$	0	90	35 (10^{-3})	40	30
PCA	24	**35** $(10^{-2}, 0.3, 0.2)$	10	60	**35** (10^{0})	10	60
F-score	500	**55** $(10^{1}, 0.3, 0.2)$	10	100	40 (10^{-2})	50	30

Table 4.11: Performance comparison of USVM with SVM is shown based on classification accuracy (%) for VolBM features. The optimal parameters for RFE process using SVM and USVM are shown in parentheses.

Features	Number of features	USVM Accuracy ($c = c_u, \epsilon, u$)	USVM Sensitivity	USVM Specificity	SVM Accuracy (c)	SVM Sensitivity	SVM Specificity
(a) CN vs AD							
CT	34	**75** $(10^{-2}, 0.7, 0.45)$	50	100	**75** (10^{-2})	50	100
SCV	23	**90** $(10^{-2}, 0.6, 0.1)$	90	90	**90** (10^{-2})	90	90
WMV	34	**50** $(10^{1}, 0.6, 0.45)$	50	50	40 (10^{2})	30	50
CT + SCV + WMV	91	**85** $(10^{-2}, 0.6, 0.3)$	70	100	80 (10^{-2})	60	100
(b) CN vs MCI							
CT	34	**68.42** $(10^{-2}, 0.6, 0.1)$	77.78	60	**68.42** (10^{-2})	77.78	60
SCV	23	**78.95** $(10^{0}, 0.8, 0.45)$	88.89	70	**78.95** (10^{0})	88.89	70
WMV	34	**84.21** $(10^{0}, 0.8, 0.3)$	100	70	78.95 (10^{0})	100	60
CT + SCV + WMV	91	**84.21** $(10^{0}, 0.6, 0.1)$	88.89	80	**84.21** (10^{0})	88.89	80
(c) MCI vs AD							
CT	34	**57.89** $(10^{-2}, 0.6, 0.3)$	40	77.78	**57.89** (10^{-2})	30	88.89
SCV	23	**73.68** $(10^{-2}, 0.6, 0.45)$	80	66.67	68.42 (10^{-2})	80	55.56
WMV	34	**63.16** $(10^{1}, 0.6, 0.45)$	60	66.67	57.89 (10^{-1})	40	77.78
CT + SCV + WMV	91	**63.16** $(10^{-2}, 0.6, 0.3)$	50	77.78	**63.16** (10^{-2})	50	77.78

4.2.6.3 VolBM features

For comprehensive analysis of MRI data, VolBM analysis is also used in this work for extracting features from the MRI images. Experiments are performed on volumetric and thickness measures obtained from Freesurfer toolbox. Table 4.11 shows the accuracy and optimal parameters of USVM and SVM on classification of CN vs AD, CN vs MCI, and MCI vs AD on VolBM features. The set of all VolBM features i.e., CT, SCV, and WMV are used for feature selection. It is clearly visible from Table 4.11 that for all the VolBM features, USVM outperforms SVM in terms of accuracy.

The accuracy of USVM for CN vs AD classification for different feature sets of RFE process is shown in Fig. 4.15. One can see that accuracy of proposed USVM-RFE is better than SVM-RFE. Moreover, the variation in accuracy of proposed USVM-RFE is less than SVM-RFE. This is the result of universum data, which helps the classifier to follow the data distribution. The classification accuracy and F1 scores on reduced

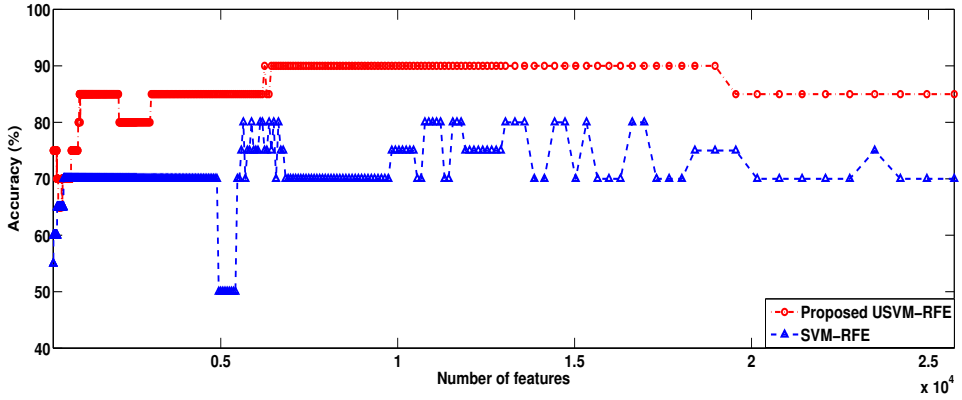
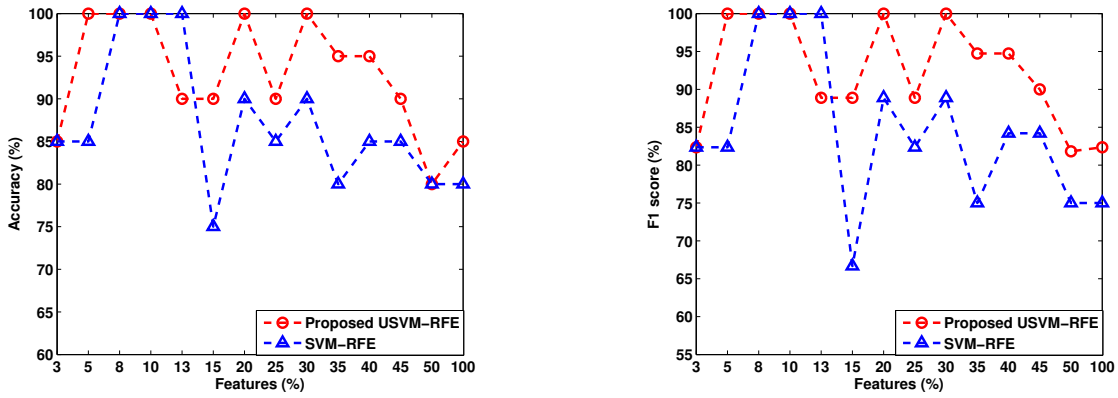


Figure 4.15: Comparison of classification accuracy of SVM-RFE and proposed USVM-RFE on different feature sets in RFE of VolBM features for CN vs AD.



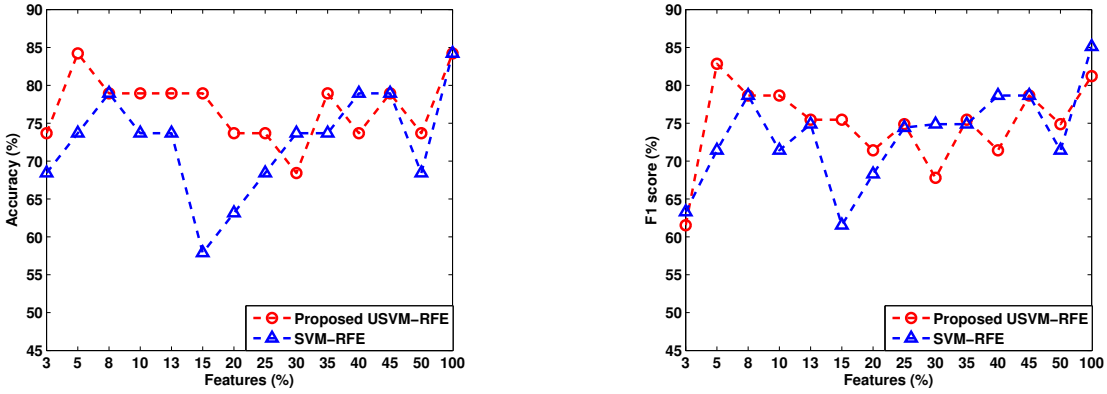
(a) (b)

Figure 4.16: Plot showing comparison of (a) accuracy, and (b) F1 score of SVM-RFE, and proposed USVM-RFE for classification of CN vs AD using VolBM features.

VolBM features using optimal parameters are shown in Figs. 4.16, 4.17, and 4.18. One may observe in Fig. 4.16 (a) that USVM-RFE achieves 100% accuracy in 5 feature sets for CN vs AD, with lowest feature set having 5% of total features.

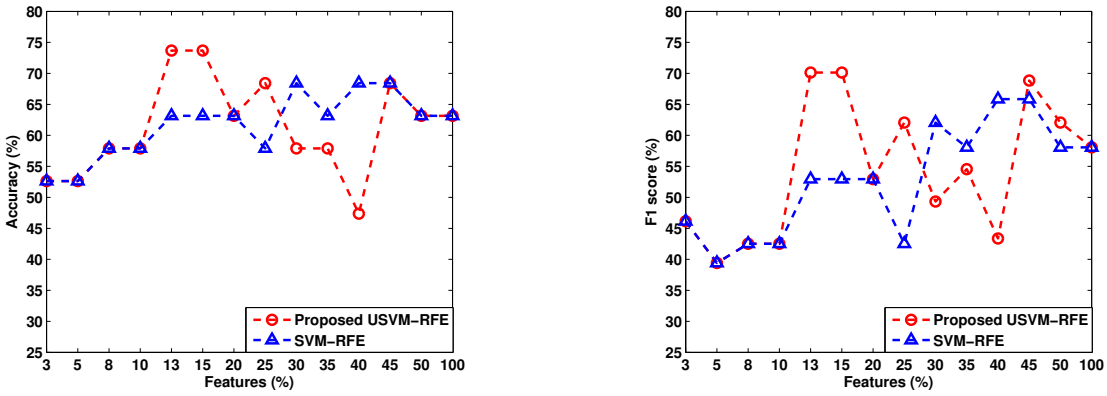
Also, the variation of accuracy looks similar in both algorithms with higher accuracy in proposed USVM-RFE. The F1 scores follow similar trend as accuracy in Fig. 4.16 (b). For CN vs MCI in Fig. 4.17 (a) proposed USVM-RFE achieves highest accuracy of 84.21% for 5% features, while SVM-RFE requires 100% features for achieving the same accuracy. Fig. 4.17 (b) shows the corresponding F1 scores.

In case of MCI vs AD in Fig. 4.18, the accuracy of proposed USVM-RFE is not



(a) (b)

Figure 4.17: Plot showing comparison of (a) accuracy, and (b) F1 score of SVM-RFE, and proposed USVM-RFE for classification of CN vs MCI using VolBM features.



(a) (b)

Figure 4.18: Plot showing comparison of (a) accuracy, and (b) F1 score of SVM-RFE, and proposed USVM-RFE for classification of MCI vs AD using VolBM features.

better than SVM-RFE in all the cases. This is the result of non-linear nature of MCI vs AD data shown in Fig. 4.19, leading to generation of improper universum data not lying between the classes. However, the proposed USVM-RFE has shown highest accuracy of 73.68% for MCI vs AD with 13% and 15% features. This is better than SVM-RFE which achieved highest accuracy of 68.42% on 30% features.

We present a formula for calculating score for feature ranking in RFE based meth-

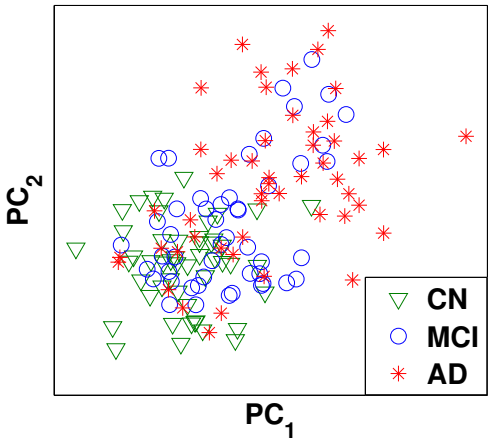


Figure 4.19: Plot of first two PCA components in VolBM features of CN, MCI, and AD classes.

ods. The feature score of the i^{th} feature is calculated as per the following,

$$Score\ (i) = \sum_{j=1}^{f} (r_{lowest} - r_{ij}), \tag{4.3}$$

where r_{ij} is the rank of i^{th} feature in j^{th} feature set, r_{lowest} is the lowest rank value in largest feature set, and f is the total number of features. By using the above mentioned formula, more weightage is given to features which survive up to the smallest feature sets. A discussion on these scores is presented in section 4.2.7.2.

4.2.6.4 ADNI baseline dataset

In order to verify the comparative performance of the proposed USVM-RFE with existing algorithms, we conducted experiments on ADNI baseline dataset [196, 197]. The VolBM features are used to compare the classification performance of the proposed USVM-RFE with SVM-RFE and TWSVM-RFE.

The comparative performance of the proposed USVM-RFE for classification CN vs AD is shown in Table 4.12. It is clearly observable that the proposed USVM-RFE is performing better than existing algorithms in most of the feature sets. The highest accuracy obtained by USVM-RFE is 89.2% with a sensitivity of 84.87% for 15% features.

Table 4.12: Comparison of performance of the proposed USVM-RFE in terms of accuracy (%) with existing algorithms on ADNI baseline dataset for CN vs AD using VolBM features.

Tissue features (%)	Proposed USVM-RFE			SVM-RFE			TWSVM-RFE		
	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
CN vs AD									
3	83.6	71.43	94.66	83.2	72.27	93.13	**84**	72.27	94.66
5	**86.4**	78.99	93.13	**86.4**	78.99	93.13	85.2	82.35	87.79
8	**84.4**	74.79	93.13	82.8	71.43	93.13	**84.4**	78.15	90.08
10	**86.4**	78.99	93.13	84.8	77.31	91.6	85.2	77.31	92.37
13	86.8	80.67	92.37	**87.2**	79.83	93.89	85.2	80.67	89.31
15	**89.2**	84.87	93.13	86.8	78.15	94.66	83.6	78.99	87.79
20	**84.4**	76.47	91.6	87.2	78.99	94.66	83.2	73.95	91.6
25	**86.8**	77.31	95.42	85.2	77.31	92.37	81.6	82.35	80.92
30	**86.4**	78.15	93.89	**86.4**	74.79	96.95	82	73.95	89.31
35	**88.8**	83.19	93.89	85.2	73.95	95.42	81.2	73.95	87.79
40	**86.4**	74.79	96.95	84.8	73.11	95.42	78.8	79.83	77.86
45	**86.8**	74.79	97.71	84.4	71.43	96.18	79.6	77.31	81.68
50	**86**	74.79	96.18	85.6	74.79	95.42	79.2	78.15	80.15

Table 4.13: Comparison of performance of the proposed USVM-RFE in terms of accuracy (%) with existing algorithms on ADNI baseline dataset for CN vs MCI using VolBM features.

Tissue features (%)	Proposed USVM-RFE			SVM-RFE			TWSVM-RFE		
	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
CN vs MCI									
3	**72.61**	86.67	47.79	69.41	75.83	58.09	71.28	67.08	78.68
5	**72.34**	84.58	50.74	71.28	86.25	44.85	69.68	86.25	40.44
8	69.41	82.08	47.06	**71.01**	86.25	44.12	69.95	82.5	47.79
10	**70.74**	76.25	61.03	70.21	82.92	47.79	70.21	82.5	48.53
13	**70.74**	81.25	52.21	69.95	80.83	50.74	**70.74**	79.17	55.88
15	**72.07**	82.92	52.94	69.95	80	52.21	68.35	75.83	55.15
20	**72.34**	81.25	56.62	70.74	82.5	50	68.62	73.75	59.56
25	70.74	85.42	44.85	**71.54**	84.17	49.26	68.09	73.33	58.82
30	69.41	82.92	45.59	**70.74**	87.08	41.91	67.82	86.67	34.56
35	68.88	73.33	61.03	**70.74**	86.25	43.38	66.49	78.75	44.85
40	**69.41**	76.25	57.35	**69.41**	86.67	38.97	67.02	62.08	75.74
45	**70.21**	78.75	55.15	69.41	86.25	39.71	68.09	76.67	52.94
50	**71.28**	80.42	55.15	68.62	85.42	38.97	68.88	81.25	47.06

In case of CN vs MCI in Table 4.13, again the proposed USVM-RFE outperfomed SVM-RFE and TWSVM-RFE in most of the feature sets. Moreover, the highest accuracy of proposed USVM-RFE i.e. 72.61% is obtained for just 3% of total features. Also, for MCI vs AD in Table 4.14, proposed USVM-RFE performed better than other algorithms in many of the feature sets. However, the highest accuracy obtained by proposed USVM-RFE is same as TWSVM-RFE i.e., 71.88%, but with higher sensitivity. However, USVM-RFE is having higher accuracy than TWSVM-RFE in lesser sized feature sets.

The results for the classification performance of USVM, SVM and TWSVM for all the features are shown in Table 4.15. One can observe that USVM is performing better

Table 4.14: Comparison of performance of the proposed USVM-RFE in terms of accuracy (%) with existing algorithms on ADNI baseline dataset for MCI vs AD using VolBM features.

Tissue features (%)	Proposed USVM-RFE			SVM-RFE			TWSVM-RFE		
	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
MCI vs AD									
3	**70.45**	36.7	85.6	68.47	24.77	88.07	69.03	0	100
5	**70.17**	38.53	84.36	69.32	55.05	75.72	63.64	64.22	63.37
8	70.17	40.37	83.54	69.6	39.45	83.13	**70.74**	17.43	94.65
10	**71.88**	41.28	85.6	71.59	40.37	85.6	**71.88**	25.69	92.59
13	**71.59**	42.2	84.77	68.18	42.2	79.84	68.75	34.86	83.95
15	**70.17**	46.79	80.66	68.47	44.95	79.01	67.05	33.95	81.89
20	**70.74**	54.13	78.19	69.89	40.37	83.13	63.92	53.21	68.72
25	**71.02**	55.05	78.19	66.76	40.37	78.6	62.5	51.38	67.49
30	**68.75**	54.13	75.31	67.33	38.53	80.25	60.23	52.29	63.79
35	**69.89**	52.29	77.78	67.05	39.45	79.42	62.5	54.13	66.26
40	**70.45**	52.29	78.6	67.61	39.45	80.25	65.34	49.54	72.43
45	**71.02**	54.13	78.6	68.18	44.95	78.6	65.06	52.29	70.78
50	**70.74**	49.54	80.25	68.18	48.62	76.95	61.65	49.54	67.08

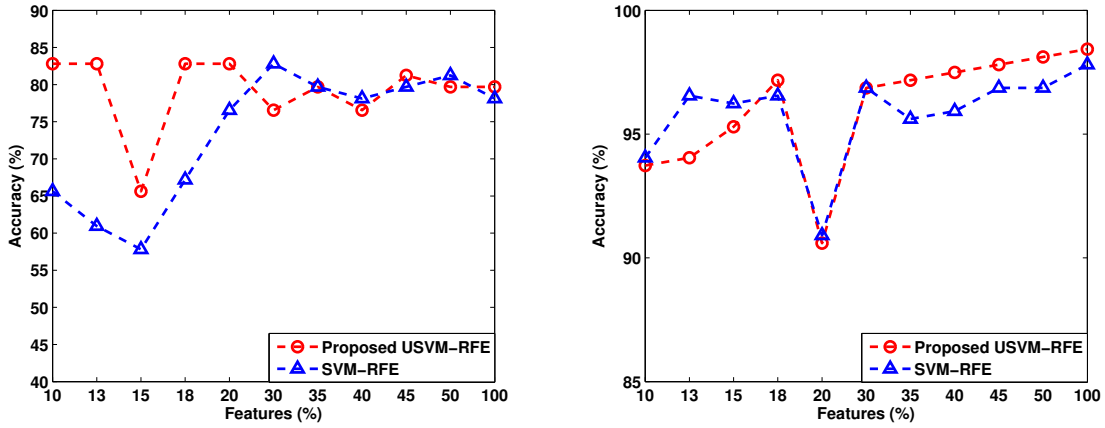
Table 4.15: Comparison of performance of USVM in terms of accuracy (%) with existing algorithms on ADNI baseline dataset using all VolBM features.

Dataset (Train size, Test size)	USVM			SVM			TWSVM		
	Accuracy ($c = c_u, \epsilon, u$)	Sensitivity	Specificity	Accuracy (c)	Sensitivity	Specificity	Accuracy (c)	Sensitivity	Specificity
CN vs AD ($165 \times 91, 250 \times 91$)	**87.6** ($10^{-2}, 0.6, 0.3$)	81.51	93.13	87.2 (10^{-2})	78.15	95.42	68 (10^{-1})	77.31	59.54
CN vs MCI ($250 \times 91, 376 \times 91$)	**70.21** ($10^{-2}, 0.8, 0.3$)	81.25	50.74	68.35 (10^{-2})	83.75	41.18	63.83 (10^{1})	68.75	55.15
MCI vs AD ($233 \times 91, 352 \times 91$)	**72.73** ($10^{-2}, 0.6, 0.3$)	39.45	87.65	64.77 (10^{0})	45.87	73.25	58.24 (10^{-6})	60.55	57.2

than the other algorithms for all the cases. This is due to the introduction of universum samples, providing prior information about AD data. The optimal parameters used in the RFE process are selected from Table 4.15.

4.2.6.5 UCI datasets

Experimental results on classification of cancer patients in Wpbc and Wdbc are shown in Fig. 4.20. The optimal parameters for RFE process are shown in Table 4.16. One can notice in Fig. 4.20(a) that the proposed USVM-RFE performs better than SVM-RFE for lower dimensional feature sets. Also, it can be observed that USVM-RFE recovers to 82.81% accuracy for 13% features immediately after a decline to accuracy of 65.63%. This is better than SVM-RFE for the same feature sets. Similarly, better performance is shown by USVM-RFE for Wdbc dataset in Fig. 4.20(b). The



(a) (b)

Figure 4.20: Plot showing comparison of classification accuracy of SVM-RFE, and proposed USVM-RFE for (a) Wpbc, and (b) Wdbc datasets.

comparison of training time is shown in Table 4.16. It can be seen that USVM takes more time than SVM. This is due to the inclusion of universum data. The addition of universum data leads to better generalization performance. So, the training time is a trade off for classification accuracy.

Table 4.16: Performance comparison of USVM with SVM is shown based on classification accuracy (%) for UCI datasets.

Dataset (Train size, Test size)	**USVM** Accuracy (c, c_u, ϵ, u) Time (s)	**SVM** Accuracy (c) Time (s)
Wpbc (130 × 33, 64 × 33)	**79.69** (10, 1, 0.5, 0.3) 0.3644	78.13 (1) 0.1396
Wdbc (250 × 30, 319 × 30)	**98.43** (0.01, 0.001, 0.5, 0.3) 1.3505	97.81 (1) 0.5170

4.2.7 Discussion

In this section, we present the discussion on reduced features obtained from VBM and VolBM analysis by SVM-RFE, and proposed USVM-RFE for CN vs AD, CN vs

MCI, and MCI vs AD cases. The different regions of brain affected in Alzheimer's disease are illustrated in Fig. 4.21.

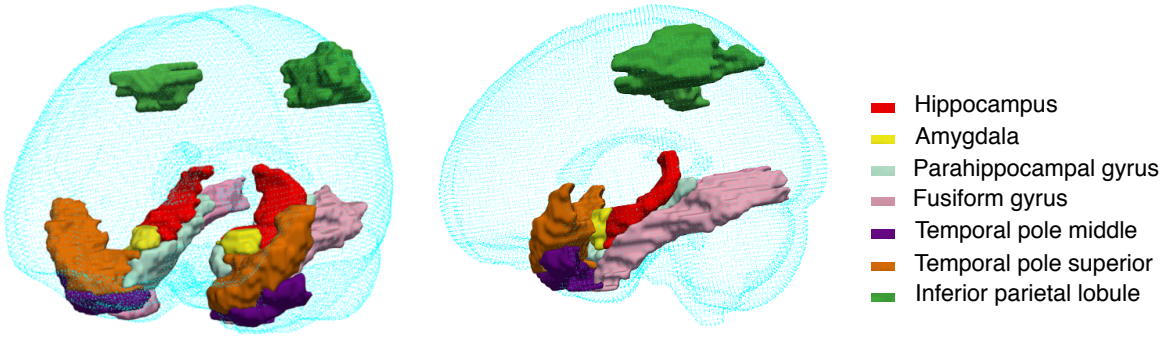
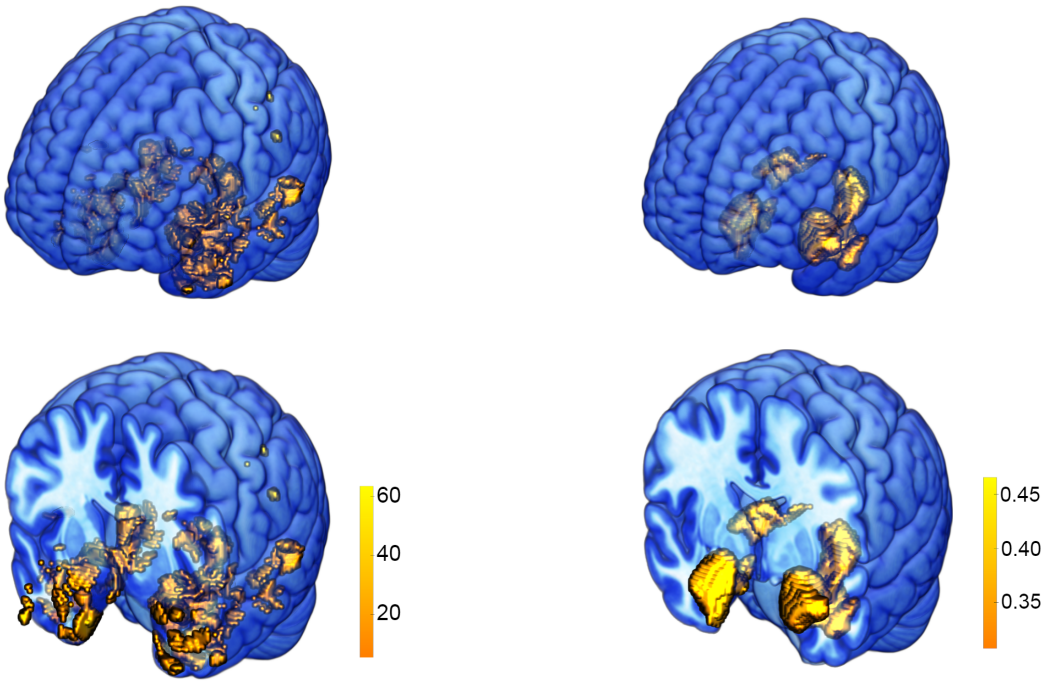


Figure 4.21: Regions of brain affected in Alzheimer's disease (AAL atlas).

4.2.7.1 VBM feature selection

In case of CN vs AD, the proposed USVM-RFE has shown better classification accuracy for less number of voxels as compared to SVM-RFE. The feature set with least size and highest accuracy is illustrated in Fig. 4.22 for both the algorithms. For SVM-RFE, the most overlapping GM regions with AAL atlas from Fig. 4.23 are: amygdala (left), amygdala (right), hippocampus (left), parahippocampal (left), parahippocampal (right), and hippocampus (right). For proposed USVM-RFE, the regions from Fig. 4.24 are: amygdala (left), hippocampus (left), parahippocampus (left), parahippocampus (right), and temporal pole mid (left). This is in accordance with previous studies [212].

For CN vs MCI, Fig. 4.25 shows the regions of increased CSF. The brain regions proximal to area of increased CSF are given in Fig. 4.26 and 4.27 for SVM-RFE and proposed USVM-RFE respectively. The regions selected by SVM-RFE are: cingulate anterior (left), temporal pole mid (left), hippocampus (left), temporal pole superior (right), and parahippocampal (right). For proposed USVM-RFE, the regions are: cingulate anterior (left), temporal pole superior (right), temporal pole mid (left), temporal pole mid (right), and parahippocampal (right). This is a result of atrophy in the temporal regions [213]. The CSF volumes of our cohort also support this finding. The mean and standard deviation of CSF volume (mm^3) of CN, MCI, and AD



(a) (b)

Figure 4.22: Illustration of reduced GM voxels (VBM) obtained after feature elimination process of (a) proposed USVM-RFE and (b) SVM-RFE for CN vs AD. The variation of weights is shown using heat map.

are 1306.43±410.93, 1493.98±388.67 and 1652.78±634.80 respectively. This indicates that CSF voxels are useful for classification of CN vs MCI. Moreover, the number of CSF voxels selected by proposed USVM-RFE for highest accuracy are lesser than SVM-RFE. This helps in localizing the regions with change in CSF volume.

In MCI vs AD, there is significant atrophy in the left temporal lobe as compared to right in both SVM-RFE and USVM-RFE features as shown in Fig. 4.28. The regions for voxels selected by SVM-RFE are: amygdala (left), parahippocampal (left), fusiform (left), hippocampus (left), temporal pole superior (left), and temporal inferior (left) (Refer Fig. 4.29 for details). Proposed USVM-RFE selected fewer voxels with higher classification accuracy as compared to SVM-RFE. The corresponding regions are: amygdala (left), parahippocampal (left), fusiform (left), temporal pole superior (left), and hippocampus (left) (Refer Fig. 4.30). Moreover, voxels selected by proposed USVM-RFE are mostly in the left side of brain. This is due to asymmetric atrophy of left and right side of brain in AD [214, 215]. The variation of amygdala

x,y,z mm	Label	% Cluster	Nb Vx Cluster	% Label	Nb Vx Label
-23 -23 -29	Hippocampus_L	25.39	4785	55.00	932
	ParaHippocampal_L	25.20	4785	52.02	978
	OUTSIDE	17.37	4785	0.00	0
	Fusiform_L	11.45	4785	10.01	2310
	Amygdala_L	9.86	4785	90.51	220
	Temporal_Pole_Sup_L	4.87	4785	7.65	1285
	Temporal_Inf_L	4.83	4785	3.05	3200
	Cerebelum_4_5_L	0.42	4785	0.75	1125
	Thalamus_L	0.38	4785	0.69	1100
	Lingual_L	0.10	4785	0.10	2095
	Putamen_L	0.08	4785	0.17	1009
	Olfactory_L	0.04	4785	0.30	280
32 -6 -21	ParaHippocampal_R	40.83	2437	37.08	1132
	Amygdala_R	17.07	2437	70.77	248
	Hippocampus_R	16.62	2437	18.06	946
	OUTSIDE	14.77	2437	0.00	0
	Fusiform_R	4.64	2437	1.89	2518
	Temporal_Pole_Sup_R	3.00	2437	2.30	1338
	Temporal_Pole_Mid_R	1.68	2437	1.46	1187
	Cerebelum_4_5_R	1.35	2437	1.62	861
	Cerebelum_3_R	0.04	2437	0.20	207
23 -33 -5	Hippocampus_R	65.37	361	10.52	946
	OUTSIDE	29.92	361	0.00	0
	ParaHippocampal_R	4.71	361	0.63	1132
6 -21 12	Thalamus_R	86.22	370	12.73	1057
	Thalamus_L	10.54	370	1.50	1100
	OUTSIDE	3.24	370	0.00	0

Figure 4.23: Reduced features obtained from SVM-RFE for CN vs AD with GM (VBM) features.

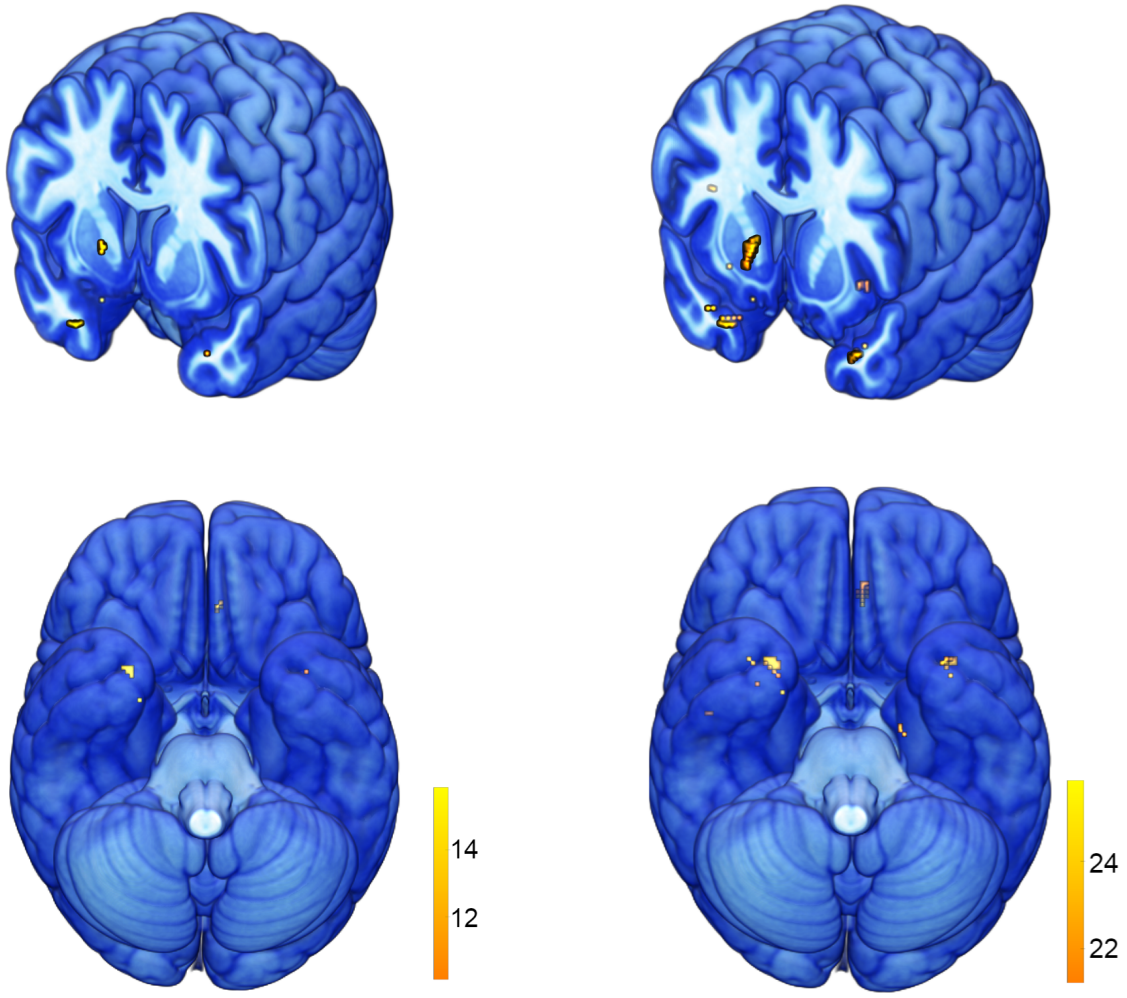
and hippocampus volume in left and right side of brain in our cohort is shown in Fig. 4.31.

In most of the cases, GM features provided better accuracy due to significant atrophy of GM regions in the brain [216]. Fig. 4.32 illustrates the variation of accuracy with feature set for SVM-RFE and proposed USVM-RFE using GM. It can be seen that the proposed USVM-RFE selects features from all the regions of brain as compared to SVM-RFE for CN vs AD.

This shows that SVM-RFE is having local information about the dataset in every iteration, while proposed USVM-RFE has global information i.e., distribution of data. Therefore, the selection of discriminative voxels from different regions of brain is achieved by USVM-RFE. It can be deduced that for classification of Alzheimer's disease, optimal features are needed from different regions of the brain.

x,y,z mm	Label	% Cluster	Nb Vx Cluster	% Label	Nb Vx Label
-23 -23 -29	Fusiform_L	49.42	172	1.55	2310
	ParaHippocampal_L	42.44	172	3.15	978
	Cerebelum_4_5_L	6.40	172	0.41	1125
	OUTSIDE	1.74	172	0.00	0
-30 3 -21	Amygdala_L	31.77	872	53.12	220
	Hippocampus_L	30.16	872	11.90	932
	ParaHippocampal_L	17.55	872	6.60	978
	OUTSIDE	15.94	872	0.00	0
	Temporal_Pole_Sup_L	3.67	872	1.05	1285
	Fusiform_L	0.92	872	0.15	2310
32 -6 -23	Hippocampus_R	75.00	8	0.27	946
	OUTSIDE	25.00	8	0.00	0
24 3 -26	OUTSIDE	46.26	294	0.00	0
	ParaHippocampal_R	24.49	294	2.68	1132
	Fusiform_R	18.03	294	0.89	2518
	Temporal_Pole_Mid_R	10.20	294	1.07	1187
	Amygdala_R	1.02	294	0.51	248
-23 -33 -3	Hippocampus_L	65.81	313	9.32	932
	OUTSIDE	22.04	313	0.00	0
	ParaHippocampal_L	10.86	313	1.47	978
	Lingual_L	0.96	313	0.06	2095
	Thalamus_L	0.32	313	0.04	1100
30 -6 -17	Amygdala_R	100.00	4	0.68	248
-38 -14 -35	Fusiform_L	52.44	82	0.79	2310
	Temporal_Inf_L	28.05	82	0.30	3200
	OUTSIDE	19.51	82	0.00	0
24 -33 -5	Hippocampus_R	53.26	92	2.19	946
	ParaHippocampal_R	25.00	92	0.86	1132
	OUTSIDE	18.48	92	0.00	0
	Lingual_R	3.26	92	0.06	2300
27 -29 -26	Cerebelum_4_5_R	58.82	85	2.45	861
	ParaHippocampal_R	29.41	85	0.93	1132
	Fusiform_R	9.41	85	0.13	2518
	OUTSIDE	2.35	85	0.00	0
-42 3 -20	Temporal_Mid_L	63.08	65	0.35	4942
	Temporal_Pole_Sup_L	27.69	65	0.59	1285
	OUTSIDE	9.23	65	0.00	0
-39 -2 -38	Temporal_Inf_L	100.00	35	0.46	3200
-35 3 -32	Temporal_Pole_Mid_L	42.68	157	3.74	755
	Temporal_Mid_L	33.12	157	0.44	4942
	Temporal_Pole_Sup_L	15.92	157	0.82	1285
	OUTSIDE	7.01	157	0.00	0
	Temporal_Inf_L	1.27	157	0.03	3200
-29 -12 -12	Hippocampus_L	43.07	274	5.34	932
	OUTSIDE	39.78	274	0.00	0
	Temporal_Sup_L	8.03	274	0.40	2296

Figure 4.24: Region of interest obtained from the proposed USVM-RFE for CN vs AD with GM (VBM) features.



(a) (b)

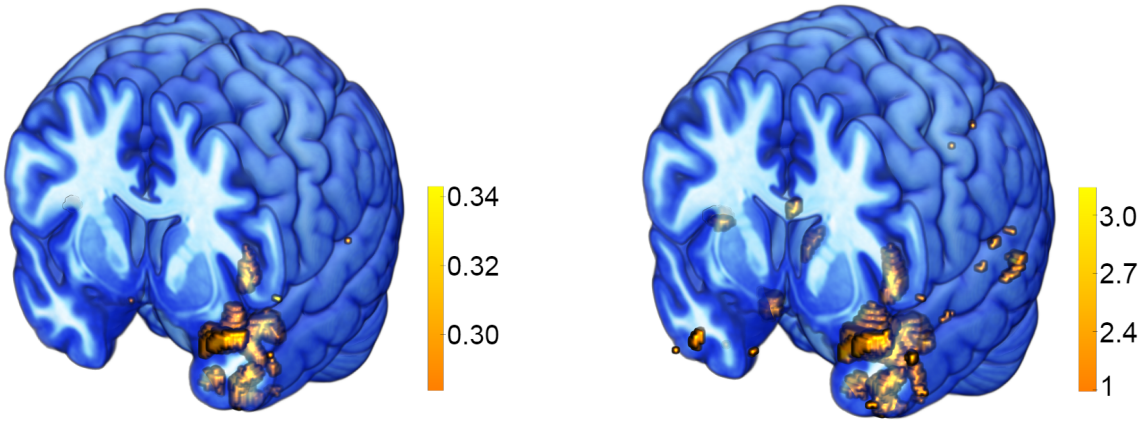
Figure 4.25: Illustration of reduced CSF voxels (VBM) obtained after feature elimination process of (a) proposed USVM-RFE and (b) SVM-RFE for CN vs MCI. The variation of weights is shown using heat map.

x,y,z mm	Label	% Cluster	Nb Vx Cluster	% Label	Nb Vx Label
27 9 -35	Temporal_Pole_Sup_R	58.33	12	0.22	1338
	ParaHippocampal_R	33.33	12	0.15	1132
	Temporal_Pole_Mid_R	8.33	12	0.04	1187
-6 39 8	Cingulate_Ant_L	100.00	33	0.99	1400
-20 -15 -14	Hippocampus_L	100.00	7	0.32	932
-35 11 -32	Temporal_Pole_Mid_L	72.73	11	0.45	755
	Temporal_Pole_Sup_L	27.27	11	0.10	1285
50 -8 9	Rolandic_Oper_R	100.00	2	0.06	1331
36 12 -30	Temporal_Pole_Sup_R	100.00	2	0.06	1338
-36 6 -29	Temporal_Pole_Sup_L	100.00	1	0.03	1285
33 3 -15	OUTSIDE	100.00	1	0.00	0
24 0 -27	ParaHippocampal_R	100.00	1	0.04	1132

Figure 4.26: Region of interest obtained from SVM-RFE for CN vs MCI with CSF (VBM) features.

x,y,z mm	Label	% Cluster	Nb Vx Cluster	% Label	Nb Vx Label
27 9 -35	Temporal_Pole_Sup_R	83.33	6	0.16	1338
	Temporal_Pole_Mid_R	16.67	6	0.04	1187
-6 36 12	Cingulate_Ant_L	100.00	6	0.18	1400
-38 11 -30	Temporal_Pole_Mid_L	100.00	1	0.06	755
24 0 -27	ParaHippocampal_R	100.00	1	0.04	1132

Figure 4.27: Region of interest obtained from USVM-RFE for CN vs MCI with CSF (VBM) features.



(a) (b)

Figure 4.28: Illustration of reduced GM voxels (VBM) obtained after feature elimination process of (a) proposed USVM-RFE and (b) SVM-RFE for MCI vs AD. The variation of weights is shown using heat map.

x,y,z mm	Label	% Cluster	Nb Vx Cluster	% Label	Nb Vx Label
-23 -23 -29	Fusiform_L	30.36	1617	8.97	2310
	OUTSIDE	19.73	1617	0.00	0
	ParaHippocampal_L	13.61	1617	9.49	978
	Hippocampus_L	11.32	1617	8.28	932
	Temporal_Pole_Sup_L	10.88	1617	5.78	1285
	Amygdala_L	10.27	1617	31.83	220
	Temporal_Inf_L	3.15	1617	0.67	3200
	Cerebelum_4_5_L	0.68	1617	0.41	1125
-23 -32 -12	ParaHippocampal_L	65.00	200	5.61	978
	OUTSIDE	16.50	200	0.00	0
	Hippocampus_L	13.00	200	1.18	932
	Lingual_L	3.00	200	0.12	2095
	Fusiform_L	2.50	200	0.09	2310
21 -33 -6	ParaHippocampal_R	35.71	42	0.56	1132
	OUTSIDE	30.95	42	0.00	0
	Lingual_R	21.43	42	0.17	2300
	Hippocampus_R	11.90	42	0.22	946
-39 -2 -38	Temporal_Inf_L	51.99	377	2.58	3200
	OUTSIDE	32.36	377	0.00	0
	Fusiform_L	11.94	377	0.82	2310
	Temporal_Pole_Mid_L	3.71	377	0.78	755
27 -15 -35	OUTSIDE	36.13	119	0.00	0
	ParaHippocampal_R	33.61	119	1.49	1132
	Fusiform_R	21.85	119	0.44	2518
	Cerebelum_4_5_R	8.40	119	0.49	861
18 3 -39	OUTSIDE	100.00	3	0.00	0
14 -15 11	Thalamus_R	100.00	26	1.04	1057
-56 3 -35	Temporal_Mid_L	47.73	44	0.18	4942
	Temporal_Inf_L	40.91	44	0.24	3200
	OUTSIDE	9.09	44	0.00	0
	Temporal_Pole_Mid_L	2.27	44	0.06	755
39 8 -36	Temporal_Pole_Mid_R	94.74	19	0.64	1187
	Temporal_Inf_R	5.26	19	0.01	3557
-66 -42 3	Temporal_Mid_L	91.67	48	0.38	4942
	OUTSIDE	8.33	48	0.00	0
69 -41 -14	Temporal_Mid_R	82.00	50	0.39	4409
	Temporal_Inf_R	18.00	50	0.11	3557
-56 6 -23	Temporal_Mid_L	87.50	16	0.12	4942
	OUTSIDE	12.50	16	0.00	0
-65 -42 15	Temporal_Sup_L	100.00	4	0.07	2296
-66 -32 14	Temporal_Sup_L	100.00	11	0.20	2296
-59 -33 2	Temporal_Mid_L	100.00	5	0.04	4942
-50 -6 -42	Temporal_Inf_L	50.00	6	0.04	3200
	OUTSIDE	50.00	6	0.00	0
54 -21 -6	Temporal_Sup_R	82.76	29	0.32	3141
	Temporal_Mid_R	17.24	29	0.05	4409

Figure 4.29: Region of interest obtained from SVM-RFE for MCI vs AD with GM (VBM) features.

x,y,z mm	Label	% Cluster	Nb Vx Cluster	% Label	Nb Vx Label
-23 -23 -29	Fusiform_L	75.64	431	5.95	2310
	ParaHippocampal_L	11.60	431	2.16	978
	OUTSIDE	8.82	431	0.00	0
	Temporal_Inf_L	2.09	431	0.12	3200
	Cerebelum_4_5_L	1.86	431	0.30	1125
-30 3 -21	Temporal_Pole_Sup_L	36.68	428	5.15	1285
	OUTSIDE	35.05	428	0.00	0
	Hippocampus_L	17.99	428	3.49	932
	Amygdala_L	10.05	428	8.25	220
	ParaHippocampal_L	0.23	428	0.04	978
-21 -32 -11	ParaHippocampal_L	76.03	121	3.97	978
	OUTSIDE	14.88	121	0.00	0
	Hippocampus_L	6.61	121	0.36	932
	Fusiform_L	2.48	121	0.05	2310
-39 -2 -38	Temporal_Inf_L	50.00	240	1.58	3200
	OUTSIDE	45.83	240	0.00	0
	Temporal_Pole_Mid_L	3.75	240	0.50	755
	Fusiform_L	0.42	240	0.02	2310
-27 -2 -39	Fusiform_L	59.38	64	0.69	2310
	Temporal_Inf_L	39.06	64	0.33	3200
	OUTSIDE	1.56	64	0.00	0
27 -20 -33	OUTSIDE	100.00	1	0.00	0
69 -41 -14	Temporal_Mid_R	100.00	14	0.13	4409
-53 -2 -33	Temporal_Inf_L	66.67	12	0.11	3200
	Temporal_Mid_L	33.33	12	0.03	4942
-50 -6 -42	Temporal_Inf_L	50.00	8	0.05	3200
	OUTSIDE	50.00	8	0.00	0
-57 6 -21	Temporal_Mid_L	100.00	1	0.01	4942
-66 -30 15	Temporal_Sup_L	100.00	1	0.02	2296
-59 3 2	Rolandic_Oper_L	100.00	2	0.09	990

Figure 4.30: Region of interest obtained from USVM-RFE for MCI vs AD with GM (VBM) features.

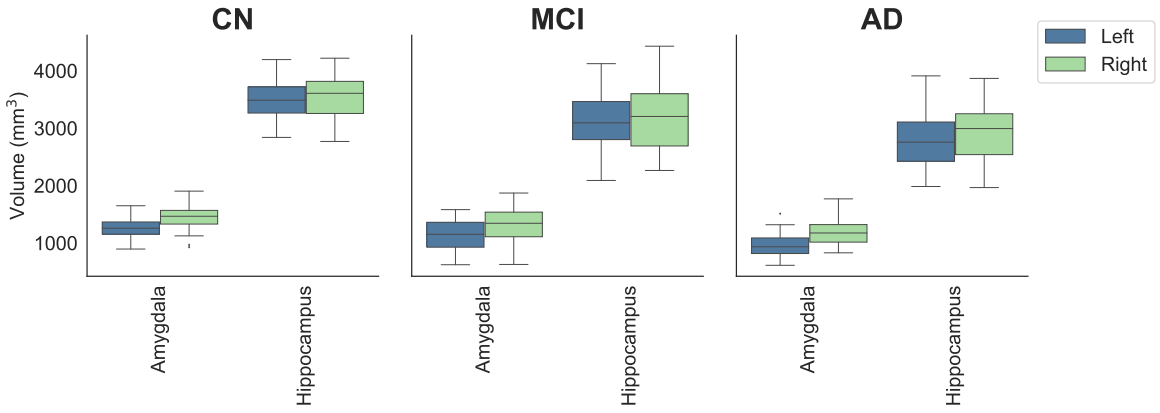


Figure 4.31: Box plot showing variation in left and right side of brain in CN, MCI, and AD.

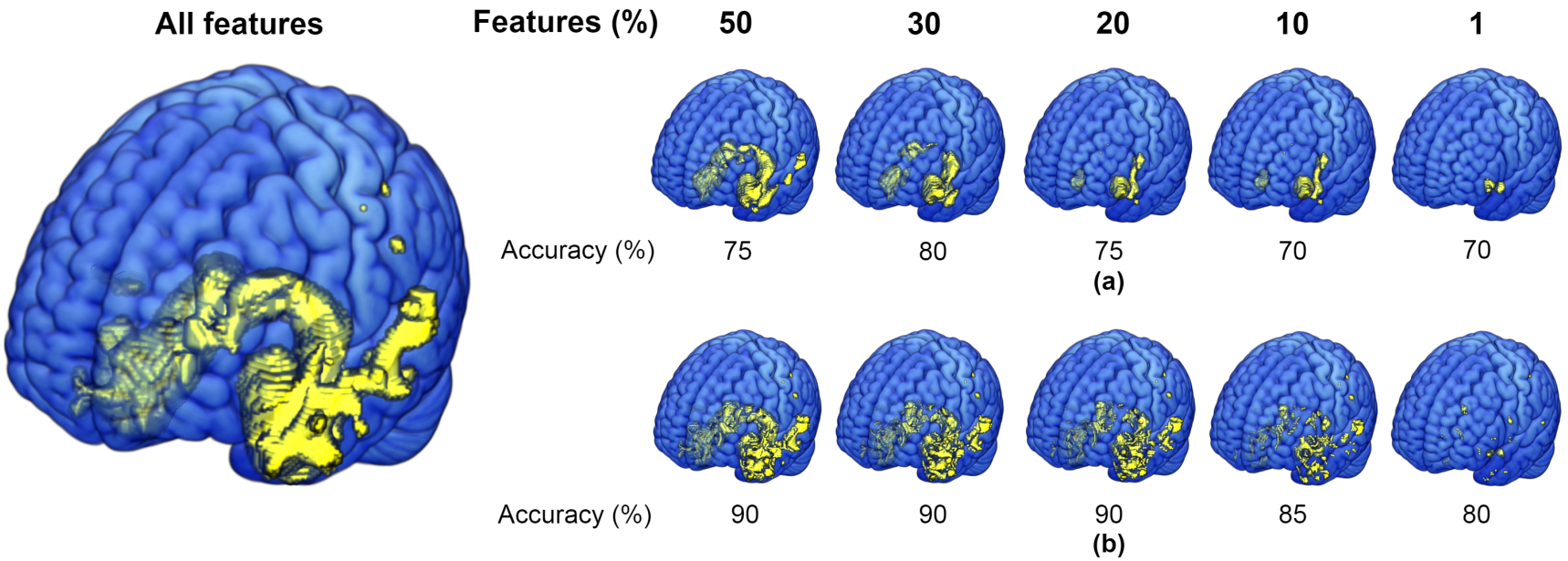


Figure 4.32: Comparison of feature elimination process based on classification accuracy of (a) SVM-RFE and (b) proposed USVM-RFE on CN vs AD for GM (VBM) features.

4.2.7.2 VolBM feature selection

The volumetric features selected by SVM-RFE and proposed USVM-RFE for CN vs AD are shown in Table 4.17. It can be seen that amygdala volume is having highest rank in both SVM-RFE and proposed USVM-RFE. This is due to atrophy of amygdala in AD subjects [217].

Other highest ranking features for SVM-RFE and USVM-RFE are parahippocampal thickness, entorhinal thickness, hippocampus volume, and inferior parietal thickness, which are in accordance with previous studies [218, 219]. This also correlates with our VBM based feature analysis. The decrease in measures of these regions is illustrated as box plots in Fig. 4.31.

However, one can see that the proposed USVM-RFE assigned higher rank to hippocampus volume than inferior parietal thickness as compared to SVM-RFE. This justifies better feature selection by proposed USVM-RFE, as hippocampus is one of the prominent features for CN vs AD classification [220]. For CN vs MCI, Table 4.18 shows that inferior parietal thickness is the most discriminative feature for SVM-RFE, while USVM-RFE ranked parahippocampal thickness as the most optimal. Other studies also suggested thinning of parahippocampal gyrus [212] in MCI patients.

For MCI vs AD, Table 4.19 shows the most significant features as superior temporal and entorhinal thickness by proposed USVM-RFE and SVM-RFE respectively. This was also shown in previous studies [219, 221]. One can observe that parahippocampal thickness is selected as discriminative feature in all cases i.e., CN vs AD, CN vs MCI, and MCI vs AD. The reason for this is the fact that parahippocampal region encloses the brain structures affected in Alzheimer's disease [222].

From the results, it is evident that the proposed USVM-RFE gives higher classification accuracy than SVM-RFE due to its prior knowledge about the distribution of data. Moreover, the features selected by proposed USVM-RFE are in accordance with the literature, justifying its effectiveness.

Table 4.17: Comparison of feature sets selected by proposed USVM-RFE with SVM-RFE from VolBM features on classification of CN vs AD. The features are represented by their feature IDs from Table 4.4.

Proposed USVM-RFE									**SVM-RFE**								
Features (%)	**13**	**10**	**8**	**5**	**3**	**Overall**	**Score**	**Feature name**	**Features (%)**	**13**	**10**	**8**	**5**	**3**	**Overall**	**Score**	**Feature name**
Accuracy (%)	90	100	100	100	85				**Accuracy (%)**	100	100	100	85	85			
Feature rank									**Feature rank**								
1	48	48	48	48	48	**48**	50	Amygdala volume	**1**	48	48	48	48	48	**48**	50	Amygdala volume
2	53	20	20	20	11	**20**	35	Parahippocampal thickness	**2**	20	20	13	20	11	**20**	35	Parahippocampal thickness
3	20	53	53	11		**11**	35	Entorhinal thickness	**3**	53	53	20	11		**11**	35	Entorhinal thickness
4	71	71	11	13		**53**	25	Hippocampus volume	**4**	11	13	53	13		**13**	29	Inferior parietal thickness
5	11	13	71			**13**	23	Inferior parietal thickness	**5**	13	12	11			**53**	23	Hippocampus volume
6	13	11	13			**71**	20	WM entorhinal volume	**6**	71	11	12			**12**	14	Fusiform thickness
7	12	12	12			**12**	12	Fusiform thickness	**7**	5	71	5			**5**	11	Temporal pole thickness
8	10	10				**10**	6	Precuneus thickness	**8**	12	5				**71**	9	WM entorhinal volume
9	5	5				**5**	4	Temporal pole thickness	**9**	57	57				**57**	4	Cortex volume
10	15					**15**	1	Isthmus cingulate thickness	**10**	8					**8**	1	Caudal middle frontal thickness
11	57					**57**	0	Cortex volume	**11**	36					**36**	0	Fourth ventricle volume

Table 4.18: Comparison of feature sets selected by proposed USVM-RFE with SVM-RFE from VolBM features on CN vs MCI classification. The features are represented by their feature IDs from Table 4.4.

Proposed USVM-RFE									**SVM-RFE**								
Features (%)	**13**	**10**	**8**	**5**	**3**	**Overall**	**Score**	**Feature name**	**Features (%)**	**13**	**10**	**8**	**5**	**3**	**Overall**	**Score**	**Feature name**
Accuracy (%)	78.95	78.95	78.95	84.21	73.68				**Accuracy (%)**	73.68	73.68	78.95	73.68	68.42			
Feature rank									**Feature rank**								
1	50	20	50	20	20	**20**	47	Parahippocampal thickness	**1**	13	13	13	13	48	**13**	49	Inferior parietal thickness
2	20	50	3	3	50	**50**	46	Cerebellum WM volume	**2**	63	50	48	48	13	**48**	42	Amygdala volume
3	34	3	20	50		**3**	31	Middle temporal thickness	**3**	40	40	40	50		**40**	31	Ventral DC volume
4	17	47	17	47		**47**	21	Accumbens area volume	**4**	48	48	50	40		**50**	30	Cerebellum WM volume
5	77	17	47			**17**	20	Lateral orbitofrontal thickness	**5**	50	63	63			**63**	21	WM middle temporal volume
6	3	77	82			**34**	15	Insula thickness	**6**	9	20	9			**9**	14	Cuneus thickness
7	13	6	34			**77**	11	WM lingual volume	**7**	20	9	20			**20**	13	Parahippocampal thickness
8	82	34				**82**	10	WM pars opercularis volume	**8**	16	67				**16**	5	Lateral occipital thickness
9	84	82				**6**	4	Transverse temporal thickness	**9**	67	16				**67**	5	WM temporal pole volume
10	47					**13**	4	Inferior parietal thickness	**10**	88					**88**	1	WM rostral anterior cingulate volume
11	6					**84**	2	WM pericalcarine volume	**11**	68					**68**	0	WM transverse temporal volume

Table 4.19: Comparison of feature sets selected by proposed USVM-RFE with SVM-RFE from VolBM features on MCI vs AD classification. The features are represented by their feature IDs from Table 4.4.

Proposed USVM-RFE										**SVM-RFE**									
Features (%)	**15**	**13**	**10**	**8**	**5**	**3**	**Overall Score**		**Feature name**	**Features (%)**	**15**	**13**	**10**	**8**	**5**	**3**	**Overall Score**		**Feature name**
Accuracy (%)	73.68	73.68	57.89	57.89	52.63	52.63				**Accuracy (%)**	63.16	63.16	57.89	57.89	52.63	52.63			
Feature rank										**Feature rank**									
1	74	74	74	4	4	4	**4**	67	Superior temporal thickness	**1**	74	11	11	74	4	4	**11**	68	Entorhinal thickness
2	11	4	11	11	11	11	**11**	65	Entorhinal thickness	**2**	11	74	4	11	11	11	**4**	65	Superior temporal thickness
3	4	11	4	74	74		**74**	56	WM lateral occipital volume	**3**	4	4	74	4	12		**74**	45	WM lateral occipital volume
4	20	68	20	68	20		**20**	43	Parahippocampal thickness	**4**	68	20	20	20	20		**20**	44	Parahippocampal thickness
5	68	20	68	20			**68**	34	WM transverse temporal volume	**5**	20	68	68	12			**12**	38	Fusiform thickness
6	12	12	48	12			**12**	27	Fusiform thickness	**6**	12	1	12	68			**68**	32	WM transverse temporal volume
7	19	19	12	2			**2**	18	Inferior temporal thickness	**7**	1	12	48	3			**3**	19	Middle temporal thickness
8	48	1	1				**48**	15	Amygdala volume	**8**	48	48	3				**1**	17	Bankssts thickness
9	2	2	2				**19**	12	Lingual thickness	**9**	3	3	1				**48**	16	Amygdala volume
10	50	48					**1**	11	Bankssts thickness	**10**	2	18					**18**	3	Medial orbitofrontal thickness
11	85	85					**85**	4	WM pars orbitalis volume	**11**	85	50					**50**	3	Cerebellum WM volume
12	1						**50**	3	Cerebellum WM volume	**12**	50						**2**	3	Inferior temporal thickness
13	82						**82**	0	WM pars opercularis volume	**13**	18						**85**	2	WM pars orbitalis volume

4.3 Summary

In this chapter, we presented two novel universum based SVM algorithms for epilepsy and Alzheimer's disease. For epilepsy, the proposed method of selection of universum points has proved to be a promising approach for the classification of healthy and seizure EEG signals. Also, the effect of outliers on the universum is reduced by using the universum from the EEG dataset itself i.e., the seizure free EEG signal. The distribution of interictal (seizure free) signals provides prior information about the distribution of healthy and seizure signals and also lies in between the two classes. Based on the experimental results, it is evident that the proposed approach using UTSVM is better in comparison to the baseline SVM based algorithms for EEG signal classification.

On the basis of our analysis on Alzheimer's disease, the proposed USVM-RFE has performed better than SVM-RFE in most cases for feature selection and classification of CN, MCI, and AD subjects. Moreover, we presented an approach of using VBM on training and testing phase separately. This is useful in real world scenarios. We provided an analysis of the feature extraction methods for MRI images i.e. voxel based and volume based features. One of the important advantages of the proposed universum based algorithm is the global or holistic approach in feature selection as compared to SVM-RFE. This provides robustness to USVM-RFE in each iteration of feature elimination.

The efficacy and better generalization performance of universum based algorithms is clearly evident from this chapter. However, to deal with noisy data in universum learning, the next chapter presents novel formulations for universum based SVM algorithms using fuzzy memberships for the data points.

Chapter 5

Fuzzy universum support vector machines

In this chapter, we present fuzzy based approaches for universum SVM algorithms. First, in section 5.1, we present fuzzy based USVM algorithms[1], termed as fuzzy universum support vector machine (FUSVM), and fuzzy universum twin support vector machine (FUTSVM). Moreover, we present a least squares based algorithm, known as fuzzy universum least squares twin SVM algorithm (FULSTSVM)[2] in section 5.2, whose solution is obtained by a system of linear equations.

5.1 Fuzzy based USVM algorithms

In 2017, Fan et al. [55] used entropy based fuzzy membership for support vector machine (EFSVM) in case of class imbalance problem. This entropy based approach is used to give higher fuzzy membership to the data points which lie at the boundary of the two classes. Motivated by this concept, we have used entropy-based fuzzy membership in the proposed FUSVM and FUTSVM. In the proposed entropy-based fuzzy approach for universum, the universum points are assigned fuzzy membership

[1]**B. Richhariya**, M. Tanveer. A fuzzy universum support vector machine based on information entropy. In M. Tanveer and Ram Bilas Pachori, editors, *Machine Intelligence and Signal Analysis*, volume 748, pages 569–582. Springer Singapore, 2019, DOI: `https://doi.org/10.1007/978-981-13-0923-6_49`.
[**Scopus Indexed**]

[2]**B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. A fuzzy universum least squares twin support vector machine (FULSTSVM). *Neural Computing and Applications*, Springer, 2021, DOI: `https://doi.org/10.1007/s00521-021-05721-4`.
[**SCI Indexed Impact Factor: 5.606**]

based on their uncertainty of belonging to any one class. The universum points are calculated using the random averaging scheme and then higher membership is assigned to the universum points based on their entropy values. The universum points which lie in between the two classes have higher entropy values as compared to the points lying nearer to one of the classes. In datasets involving noise and outliers, the universum data generated by random averaging do not lie in between the two classes. By the use of a fuzzy-based approach for the universum data, the effect of outlier universum points is reduced, leading to higher generalization performance.

5.1.1 Proposed fuzzy USVM (FUSVM)

The optimization problem of FUSVM is written as

$$\begin{aligned} \min_{w,b,\xi,\eta} \quad & \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{l}\xi_i + c_u\sum_{j=1}^{2r} f_j\eta_j \\ s.t. \quad & y_i(w^T\phi(x_i)+b) \geq 1-\xi_i, \qquad (5.1) \\ & y_j(w^T\phi(x_j)+b) \geq -\epsilon-\eta_j, \\ & \xi_i \geq 0,\ \eta_j \geq 0,\ \forall i = 1,2,\ldots,l,\ \forall j = 1,2,\ldots,2r, \qquad (5.2) \end{aligned}$$

where l is the total number of data points, $c > 0, c_u > 0$ are penalty parameters, ξ_i and η_j are slack variables, ϵ is the parameter for the ϵ-insensitive tube, $\phi : R^n \rightarrow R^p$ is the function mapping from n to p dimension where $p > n$, and r is the number of universum samples.

The dual formulation of Eq. (5.2) is written by applying the K.K.T. conditions as

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l+2r}\mu_i\alpha_i - \frac{1}{2}\sum_{i=1}^{l+2r}\sum_{j=1}^{l+2r}\alpha_i\alpha_j y_i y_j \phi(x_i)^T\phi(x_j) \\ s.t. \quad & 0 \leq \alpha_i \leq c,\ \mu_i = 1,\ \forall\, i = 1,2,\ldots,l, \\ & 0 \leq \alpha_i \leq f_i c_u,\ \mu_i = -\epsilon,\ \forall\, i = l+1, l+2,\ldots,l+2r, \\ & \sum_{i=1}^{l+2r}\alpha_i y_i = 0, \qquad (5.3) \end{aligned}$$

where $\alpha_i, \alpha_j \geq 0$ is the Lagrange multiplier.

For any data point $x \in R^n$, the classifier is written as

$$f(x) = sgn\left(\sum_{i=1}^{l+2r} \alpha_i y_i K(x_i, x) + b\right). \tag{5.4}$$

5.1.2 Proposed fuzzy universum twin support vector machine (FUTSVM)

By introducing weights to the universum data points based on their information entropy, the non-linear FUTSVM comprises the following minimization problems,

$$\begin{aligned} \min_{w_1, b_1, \xi_1, \eta_1} \quad & \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_1 + c_u f_u^T \eta_1 \\ s.t. \quad & -(K(X_2, D^T)w_1 + e_2 b_1) + \xi_1 \geq e_2, \\ & (K(U, D^T)w_1 + e_u b_1) + \eta_1 \geq (-1+\epsilon)e_u, \\ & \xi_1 \geq 0, \quad \eta_1 \geq 0, \end{aligned} \tag{5.5}$$

$$\begin{aligned} \min_{w_2, b_2, \xi_2, \eta_2} \quad & \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_2 + c_u f_u^T \eta_2 \\ s.t. \quad & (K(X_1, D^T)w_2 + e_1 b_2) + \xi_2 \geq e_1, \\ & -(K(U, D^T)w_2 + e_u b_2) + \eta_2 \geq (-1+\epsilon)e_u, \\ & \xi_2 \geq 0, \quad \eta_2 \geq 0, \end{aligned} \tag{5.6}$$

where $K(X_i, D^T)$ is the kernel matrix, $D = [X_1; X_2]$, f_u is the vector containing the fuzzy membership values, $c_i (i = 1, 2)$ and c_u are positive penalty parameters, ξ_i, $\eta_i (i = 1, 2)$ are slack variables, and $e_i (i = 1, 2)$ is a vector of ones of suitable dimension.

In comparison to UTSVM, the fuzzy-based approach of FUTSVM is helpful in reducing the effect of outliers in the universum data. In FUTSVM, appropriate membership value is given to the universum points based on their information entropy. This approach reduces the effect of noise on the universum and results in better gen-

eralization performance.

By applying the K.K.T. necessary and sufficient conditions, the Wolfe duals of Eqs. (5.5) and (5.6) are obtained as

$$\begin{aligned} &\max_{\alpha_1,\,\mu_1}\ e_2^T\alpha_1 - \frac{1}{2}(\alpha_1^T N - \mu_1^T O)(M^T M)^{-1}(N^T\alpha_1 - O^T\mu_1) + (\epsilon-1)e_u^T\mu_1 \\ &s.t.\ \ 0 \le \alpha_1 \le c_1, \quad 0 \le \mu_1 \le f_u c_u, \end{aligned} \tag{5.7}$$

$$\begin{aligned} &\max_{\alpha_2,\,\mu_2}\ e_2^T\alpha_2 - \frac{1}{2}(\alpha_2^T M - \mu_2^T O)(N^T N)^{-1}(M^T\alpha_2 - O^T\mu_2) + (\epsilon-1)e_u^T\mu_2 \\ &s.t.\ \ 0 \le \alpha_2 \le c_2, \quad 0 \le \mu_2 \le f_u c_u, \end{aligned} \tag{5.8}$$

where $M = [K(X_1,\, D^T)\ e_1]$, $N = [K(X_2,\, D^T)\ e_2]$, and $O = [K(U,\, D^T)\ e_u]$; α_1, α_2, μ_1, μ_2 are the vectors of Lagrange multipliers.

The classifying hyperplanes $K(x^T, D^T)w_1 + b_1 = 0$ and $K(x^T, D^T)w_2 + b_2 = 0$ are constructed from the parameter values of $w_i (i = 1, 2)$ and b_i using the following Eqs. (5.9) and (5.10),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(M^T M)^{-1}(N^T\alpha_1 - O^T\mu_1), \tag{5.9}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (N^T N)^{-1}(M^T\alpha_2 - O^T\mu_2). \tag{5.10}$$

To avoid the ill-conditioning in the calculation of inverse $(M^T M)^{-1}$ and $(N^T N)^{-1}$, we add a regularization term δI to the matrices in (5.9) and (5.10) as $(M^T M + \delta I)^{-1}$ and $(N^T N + \delta I)^{-1}$ to make them positive definite where δ is a small positive value. Here, I is an identity matrix of appropriate dimension.

Each new data point is classified using Eq. (2.10).

5.1.3 Calculation of fuzzy membership

In this work, we used random averaging method for selecting the universum data. However, the strategy of random averaging gives equal importance to all the universum data points. There is no knowledge about the suitability of each of these generated universum data points for use in universum learning. Therefore, a selection strategy is needed to remove the data points which are less suitable for universum learning.

Motivated by Fan et al. [55], we used a fuzzy membership approach for the universum data points in this work. In the proposed approach, the universum data points are eliminated based on their fuzzy membership values. The fuzzy membership values are calculated based on the information entropy of the universum data points. The fuzzy membership calculation based on information entropy helps to identify those points which are having highest uncertainty of belonging to one of the two classes. These are the data points with the highest information entropy. Due to this selection criteria, we get the universum data points lying in the region of high uncertainty w.r.t. their class labels. This is very much desired in universum learning, since the universum data do not belong to any of the binary classes. This is how proper information about the prior distribution is provided by universum data points. Therefore, by using the proposed approach, we obtain the universum data points which lie in between the data points of the binary classes. The universum points obtained by using this approach have been utilized in the proposed FUSVM and FUTSVM algorithms.

The fuzzy membership values for the universum data points are calculated as per the following:

(i). Calculate the information entropy of the universum data points based on K-nearest neighbour (KNN) approach using Euclidean distance. The probability value of the universum data point is calculated based on the class label of its neighbours.

(ii). Assign the universum data points to 10 subsets in decreasing order of entropy

values [55]. The formula for information entropy is as follows:

$$E = -p_+ ln(p_+) - p_- ln(p_-), \tag{5.11}$$

where p_+ and p_- are the probabilities of a data point to belong to the positive and negative class respectively, and ln is the natural logarithm.

(iii). The fuzzy membership value is calculated as

$$f(i) = 1 - \kappa(n_i - 1), \tag{5.12}$$

where $\kappa = 0.05$ is the fuzzy parameter and n_i is the i^{th} subset, $i = 1, 2, \ldots, 10$.

The proposed approach of assigning fuzzy membership based on information entropy gives more weight to those universum points which are lying in between the two classes. In case of universum data with outliers, this fuzzy-based approach is useful in order to reduce the effect of outliers on the SVM classifier.

5.1.4 Experimental results

In this section, to check the effectiveness of the proposed algorithms, we have performed numerical experiments on benchmark datasets with comparisons to the baseline methods.

For FUSVM and FUTSVM, the value of K for KNN is chosen as 5. For FUSVM, USVM, FUTSVM and TWSVM the value of ϵ is chosen from $\{0.1, 0.3, 0.5, 0.6\}$. For all the methods RBF kernel is used, and the value of μ is calculated using Eq. (4.1). To reduce the computational cost of the parameter selection, we set $c = c_1 = c_2 = c_u$ and chosen according to subsection 3.1.5 for all the algorithms. For USVM, FUSVM, UTSVM and FUTSVM, the universum is calculated using random averaging scheme, and the size of the universum is taken as 30% of the size of training data. The results are shown in Tables 5.1 and 5.2 for the proposed FUSVM and FUTSVM in comparison to existing algorithms for prediction accuracy and training time, with the corresponding ranks on accuracy.

Table 5.1: Performance comparison of proposed FUSVM with SVM and USVM.

Dataset (Train size, Test size)	**SVM** Accuracy (%) (c, μ) Time (s)	**USVM** Accuracy (%) (c, μ, ϵ) Time (s)	**Proposed FUSVM** Accuracy (%) (c, μ, ϵ) Time (s)
German $(400 \times 24, 600 \times 24)$	76 $(10^1, 6.80536)$ 0.3027	**77.5** $(10^0, 6.80536, 0.1)$ 3.7708	**77.5** $(10^0, 6.80536, 0.3)$ 3.872
Cleveland $(150 \times 13, 147 \times 13)$	80.95 $(10^1, 5.26173)$ 0.0365	81.63 $(10^0, 5.26173, 0.1)$ 0.5113	**82.99** $(10^0, 5.26173, 0.3)$ 0.5453
Ionosphere $(150 \times 33, 201 \times 33)$	**89.55** $(10^1, 4.38631)$ 0.0377	87.06 $(10^3, 4.38631, 0.1)$ 0.5274	89.05 $(10^0, 4.38631, 0.1)$ 0.546
Transfusion $(350 \times 4, 398 \times 4)$	82.41 $(10^2, 2077.88)$ 0.192	82.66 $(10^5, 2077.88, 0.3)$ 2.8532	**84.17** $(10^5, 2077.88, 0.6)$ 2.916
Cmc $(500 \times 9, 973 \times 9)$	74.2 $(10^3, 13.4139)$ 0.4126	69.37 $(10^1, 13.4139, 0.3)$ 5.779	**74.51** $(10^3, 13.4139, 0.1)$ 6.0853
Heart-stat $(180 \times 13, 90 \times 13)$	77.78 $(10^2, 85.982)$ 0.053	**81.11** $(10^2, 85.982, 0.5)$ 0.7478	80 $(10^2, 85.982, 0.5)$ 0.7644
Monk3 $(250 \times 7, 304 \times 7)$	83.22 $(10^5, 163.314)$ 0.1036	**84.21** $(10^5, 163.314, 0.1)$ 1.4406	83.88 $(10^5, 163.314, 0.1)$ 1.5038
Ndc1k $(400 \times 32, 700 \times 32)$	89.29 $(10^2, 571.157)$ 0.269	93.29 $(10^3, 571.157, 0.3)$ 3.865	**93.71** $(10^3, 571.157, 0.3)$ 3.9502
Pima-Indians $(350 \times 8, 418 \times 8)$	**80.38** $(10^3, 2.22928)$ 0.1966	77.51 $(10^4, 2.22928, 0.1)$ 2.8562	78.71 $(10^3, 2.22928, 0.5)$ 2.9653
Wdbc $(250 \times 30, 319 \times 30)$	94.98 $(10^4, 944.407)$ 0.111	95.61 $(10^5, 944.407, 0.3)$ 1.4658	**95.92** $(10^5, 944.407, 0.6)$ 1.5308
Yeast1 $(600 \times 8, 2368 \times 8)$	75.34 $(10^3, 0.41735)$ 0.6244	**76.39** $(10^0, 0.41735, 0.1)$ 8.5181	75.97 $(10^0, 0.41735, 0.1)$ 8.7554

Table 5.1 (contd.)

Dataset (Train size, Test size)	**SVM** Accuracy (%) (c, μ) Time (s)	**USVM** Accuracy (%) (c, μ, ϵ) Time (s)	**Proposed FUSVM** Accuracy (%) (c, μ, ϵ) Time (s)
Ripley $(200 \times 2, 1050 \times 2)$	90.48 $(10^0, 0.766998)$ 0.066	89.81 $(10^0, 0.766998, 0.6)$ 0.923	**91.05** $(10^4, 0.766998, 0.6)$ 0.9578
Yeast3 $(500 \times 8, 984 \times 8)$	94.31 $(10^0, 0.411172)$ 0.4341	**95.43** $(10^0, 0.411172, 0.1)$ 5.9856	95.33 $(10^0, 0.411172, 0.1)$ 6.1587
Monk2 $(150 \times 7, 451 \times 7)$	**67.63** $(10^{-5}, 151.849)$ 0.0748	58.76 $(10^1, 151.849, 0.3)$ 0.5508	**67.63** $(10^0, 151.849, 0.5)$ 0.5779
Average accuracy	82.61	82.17	**83.6**
Average rank	2.4643	2.0357	**1.5**

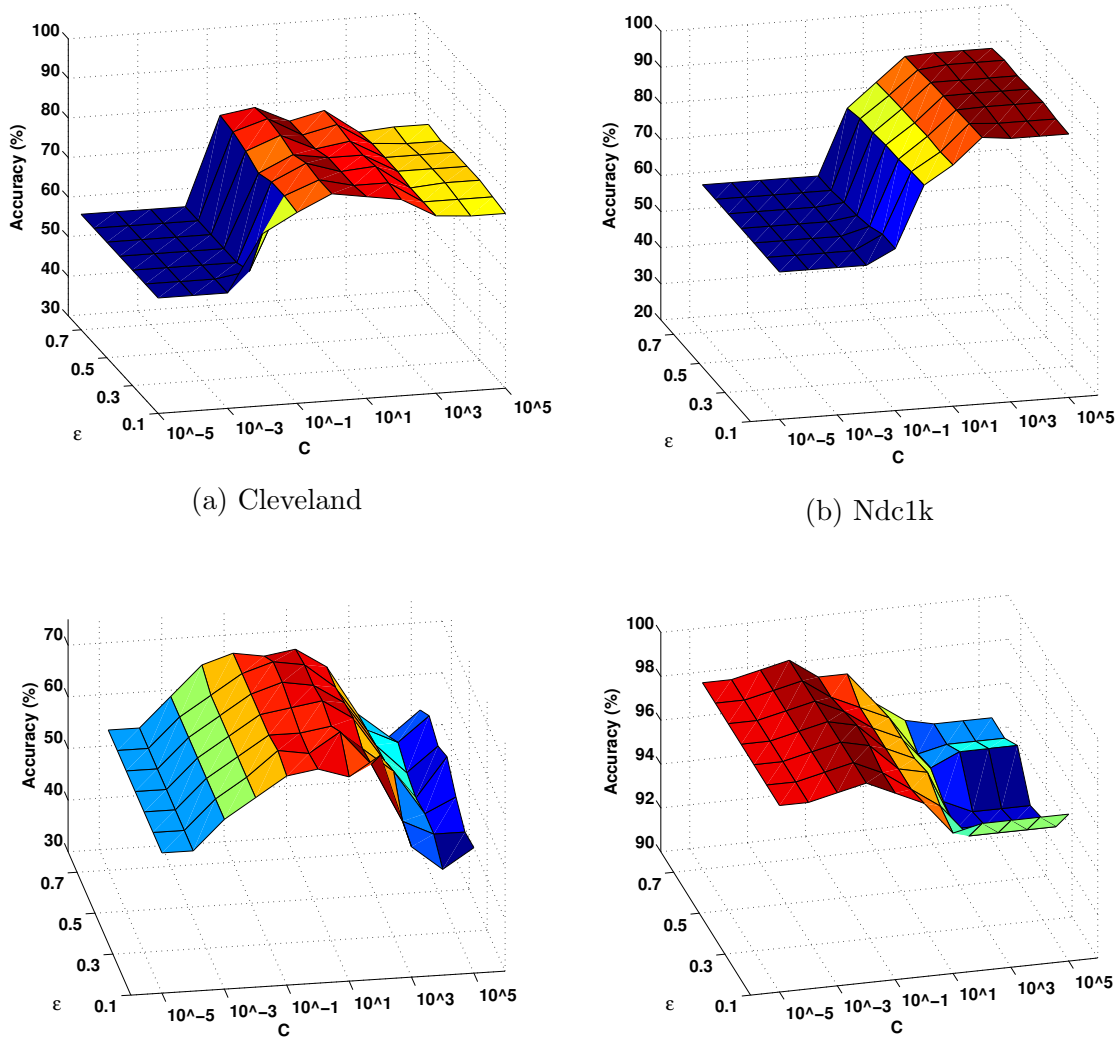
Table 5.2: Performance comparison of proposed FUTSVM with TWSVM and UTSVM.

Dataset (Train size, Test size)	**TWSVM** Accuracy (%) (c_1, μ) Time (s)	**UTSVM** Accuracy (%) (c_1, μ, ϵ) Time (s)	**Proposed FUTSVM** Accuracy (%) (c_1, μ, ϵ) Time (s)
German $(400 \times 24, 600 \times 24)$	**72** $(10^{-4}, 6.80536)$ 0.0911	**72** $(10^{-4}, 6.80536, 0.1)$ 0.131	**72** $(10^{-4}, 6.80536, 0.1)$ 0.2365
Bupa or liver disorders $(240 \times 6, 105 \times 6)$	70.48 $(10^{-1}, 66.1988)$ 0.0183	69.52 $(10^1, 66.1988, 0.6)$ 0.0357	**71.43** $(10^1, 66.1988, 0.6)$ 0.068
Cleveland $(150 \times 13, 147 \times 13)$	75.51 $(10^{-5}, 5.26173)$ 0.009	75.51 $(10^{-2}, 5.26173, 0.5)$ 0.0148	**76.19** $(10^{-2}, 5.26173, 0.6)$ 0.0274
Ionosphere $(150 \times 33, 201 \times 33)$	**92.54** $(10^{-2}, 4.38631)$ 0.0094	91.54 $(10^{-2}, 4.38631, 0.5)$ 0.0145	91.54 $(10^{-2}, 4.38631, 0.6)$ 0.0288
Transfusion $(350 \times 4, 398 \times 4)$	82.16 $(10^{-5}, 2077.88)$ 0.0387	82.41 $(10^0, 2077.88, 0.3)$ 0.0712	**82.66** $(10^0, 2077.88, 0.3)$ 0.1417

Table 5.2 (contd.)

Dataset (Train size, Test size)	**TWSVM** Accuracy (%) (c_1, μ) Time (s)	**UTSVM** Accuracy (%) (c_1, μ, ϵ) Time (s)	**Proposed FUTSVM** Accuracy (%) (c_1, μ, ϵ) Time (s)
Heart-stat $(180 \times 13, 90 \times 13)$	**81.11** $(10^{-1}, 85.982)$ 0.0121	77.78 $(10^{0}, 85.982, 0.5)$ 0.0183	80 $(10^{0}, 85.982, 0.6)$ 0.0376
Monk3 $(250 \times 7, 304 \times 7)$	57.89 $(10^{-5}, 163.314)$ 0.018	68.42 $(10^{3}, 163.314, 0.3)$ 0.0388	**78.29** $(10^{4}, 163.314, 0.3)$ 0.0782
Ndc1k $(400 \times 32, 700 \times 32)$	90.14 $(10^{-1}, 571.157)$ 0.0559	93.14 $(10^{0}, 571.157, 0.5)$ 0.103	**93.43** $(10^{0}, 571.157, 0.6)$ 0.1971
Pima-Indians $(350 \times 8, 418 \times 8)$	75.12 $(10^{0}, 2.22928)$ 0.0374	**77.27** $(10^{1}, 2.22928, 0.6)$ 0.0724	**77.27** $(10^{1}, 2.22928, 0.5)$ 0.1474
Wdbc $(250 \times 30, 319 \times 30)$	**95.92** $(10^{1}, 944.407)$ 0.0212	91.85 $(10^{-1}, 944.407, 0.6)$ 0.0362	**95.92** $(10^{0}, 944.407, 0.1)$ 0.0768
Vehicle2 $(400 \times 18, 446 \times 18)$	**98.43** $(10^{0}, 269.333)$ 0.0579	97.76 $(10^{1}, 269.333, 0.1)$ 0.1151	97.98 $(10^{1}, 269.333, 0.1)$ 0.21
Vehicle2 $(400 \times 18, 446 \times 18)$	**98.43** $(10^{0}, 269.333)$ 0.0579	97.76 $(10^{1}, 269.333, 0.1)$ 0.1151	97.98 $(10^{1}, 269.333, 0.1)$ 0.21
Yeast1 $(600 \times 8, 2368 \times 8)$	**75.84** $(10^{0}, 0.41735)$ 0.1632	75.3 $(10^{-1}, 0.41735, 0.5)$ 0.2959	75.3 $(10^{-1}, 0.41735, 0.5)$ 0.4888
Yeast3 $(500 \times 8, 984 \times 8)$	94.41 $(10^{-2}, 0.411172)$ 0.1349	93.09 $(10^{0}, 0.411172, 0.1)$ 0.247	**94.61** $(10^{-3}, 0.411172, 0.6)$ 0.3805
Breast cancer wisconsin $(350 \times 9, 333 \times 9)$	98.5 $(10^{-4}, 12.5292)$ 0.0774	**98.8** $(10^{-3}, 12.5292, 0.1)$ 0.1154	**98.8** $(10^{-3}, 12.5292, 0.6)$ 0.1935
Average accuracy	82.87	83.17	**84.67**
Average rank	2.0714	2.3929	**1.5357**

From Tables 5.1 and 5.2, it is evident that the proposed FUSVM and FUTSVM are giving better generalization performance in comparison to other baseline algorithms. It is also reflected in the average ranks based on accuracy shown in these tables. However, the proposed methods (FUSVM and FUTSVM) are taking some additional computational time due to the fuzzy calculation. This additional time can be traded for the improved generalization ability of the model.



(a) Cleveland

(b) Ndc1k

(c) Bupa or liver disorders

(d) Breast cancer wisconsin

Figure 5.1: Insensitivity performance for classification of FUSVM is shown in (a) and (b), and for FUTSVM in (c) and (d) to the user specified parameters (c, ϵ) on real world datasets using RBF kernel.

The insensitivity analysis for both FUSVM and FUTSVM is shown in Fig. 5.1

for the parameters c and ϵ. One can observe from Fig. 5.1 that FUSVM gives better accuracy for larger values of c. However, the proposed FUTSVM shows better generalization performance for lesser values of c.

The next section presents a fuzzy based approach in a least squares based twin SVM model using universum data. The algorithm is robust w.r.t. noisy data, and also requires less computation time. The proposed fuzzy based approach results in an noise insensitive as well as efficient model, which is discussed in the following section.

5.2 Proposed fuzzy universum least squares twin support vector machine (FULSTSVM)

In universum based algorithms, all the universum data points are not equally important for the classifier. To solve these problems, a novel fuzzy universum least squares twin support vector machine (FULSTSVM) is proposed in this work. In FULSTSVM, the membership values are used to provide weights for the data samples of the classes, as well as to the universum data. Further, the optimization problem of proposed FULSTSVM is obtained by solving a system of linear equations. This leads to an efficient fuzzy based algorithm.

In the next subsection, we present the formulation of the proposed FULSTSVM, and the fuzzy function used in the proposed algorithm. The proposed algorithm is motivated by the approach used in RFLSTSVM-CIL [25] for removing the effect of outliers. In proposed FULSTSVM, the fuzzy memberships are calculated for the data samples belonging to the classes, as well as to the universum using fuzzy membership matrices as described below.

5.2.1 Linear FULSTSVM

The formulation of proposed FULSTSVM for the linear case is described using optimization problems (5.13) and (5.14). In the objective function of the primal problem (5.13), we use three diagonal matrices represented by S_i containing the fuzzy

memberships of the data points of i^{th} class. The memberships of the data points are calculated on the basis of distance from their respective class centres.

We also add a regularization term in the objective function to include the structural risk minimization principle (SRM) principle. The constraints are similar to the UL-STSVM formulation described in the previous subsection. Fig. 5.2 shows a pictorial representation of the proposed approach.

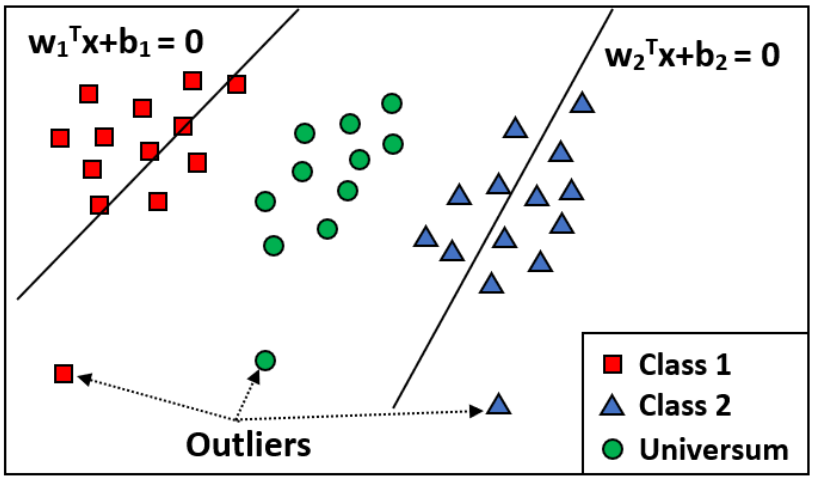


Figure 5.2: Universum data with noise.

$$\begin{aligned} \min_{w_1,b_1,\xi_1,\psi_1} \quad & \frac{1}{2}\|S_1(X_1w_1 + e_1b_1)\|^2 + \frac{c_1}{2}\|S_2\xi_1\|^2 + \frac{c_3}{2}(\|w_1\|^2 + b_1^2) + \frac{c_u}{2}\|S_u\psi_1\|^2 \\ s.t. \quad & -(X_2w_1 + e_2b_1) + \xi_1 = e_2, \\ & Uw_1 + e_ub_1 + \psi_1 = (-1+\epsilon)e_u, \end{aligned} \tag{5.13}$$

$$\begin{aligned} \min_{w_2,b_2,\xi_2,\psi_2} \quad & \frac{1}{2}\|S_2(X_2w_2 + e_2b_2)\|^2 + \frac{c_2}{2}\|S_1\xi_2\|^2 + \frac{c_4}{2}(\|w_2\|^2 + b_2^2) + \frac{c_u}{2}\|S_u\psi_2\|^2 \\ s.t. \quad & X_1w_2 + e_1b_2 + \xi_2 = e_1, \\ & -(Uw_2 + e_ub_2) + \psi_2 = (-1+\epsilon)e_u, \end{aligned} \tag{5.14}$$

where S_i, S_u are diagonal matrices containing fuzzy membership values of data samples belonging to the classes and universum respectively. ξ_i, ψ_i, are the slack variables, and

c_i, c_u are positive penalty parameters, $i = 1, 2$. The parameter for the insensitive zone is ϵ, while c_i, $i = 3, 4$ are the parameters for regularization.

Rewriting the objective functions using the values of the error variables,

$$\min_{w_1,b_1} \quad \frac{1}{2}\|S_1(X_1w_1 + e_1b_1)\|^2 + \frac{c_1}{2}\|S_2(X_2w_1 + e_2b_1 + e_2)\|^2 + \frac{c_3}{2}(\|w_1\|^2 + b_1^2) + \frac{c_u}{2}\|S_u(-(Uw_1 + e_ub_1) + (-1+\epsilon)e_u)\|^2, \tag{5.15}$$

$$\min_{w_2,b_2} \quad \frac{1}{2}\|S_2(X_2w_2 + e_2b_2)\|^2 + \frac{c_2}{2}\|S_1(-(X_1w_2 + e_1b_2) + e_1)\|^2 + \frac{c_4}{2}(\|w_2\|^2 + b_2^2) + \frac{c_u}{2}\|S_u(Uw_2 + e_ub_2 + (-1+\epsilon)e_u)\|^2. \tag{5.16}$$

By setting the gradient of QPP (5.15) w.r.t. w_1 and b_1 equal to 0, and solving we get

$$c_3w_1 + (S_1X_1)^T(S_1(X_1w_1 + e_1b_1)) + c_1(S_2X_2)^T(S_2(X_2w_1 + e_2b_1 + e_2)) - c_u(S_uU)^T(S_u(-(Uw_1 + e_ub_1) + (-1+\epsilon)e_u) = 0, \tag{5.17}$$

$$c_3b_1 + (S_1e_1)^T(S_1(X_1w_1 + e_1b_1)) + c_1(S_2e_2)^T(S_2(X_2w_1 + e_2b_1 + e_2)) - c_u(S_ue_u)^T(S_u(-(Uw_1 + e_ub_1) + (-1+\epsilon)e_u) = 0, \tag{5.18}$$

Rewriting Eqs. (5.17) and (5.18) with $u_1 = [w_1 \;\; b_1]^T$ and combining, we get

$$c_3u_1 + V^TVu_1 + c_1W^TWu_1 + c_1W^TS_2e_2 + c_uZ^TZu_1 + c_uZ^TS_u(1-\epsilon)e_u = 0. \tag{5.19}$$

Rearranging the terms and solving, we get

$$[w_1 \;\; b_1]^T = -(V^TV + c_1W^TW + c_3I + c_uZ^TZ)^{-1}(c_1W^TS_2e_2 + c_uZ^TS_u(1-\epsilon)e_u), \tag{5.20}$$

where $V = [S_1X_1 \;\; S_1e_1]$, $W = [S_2X_2 \;\; S_2e_2]$, and $Z = [S_uU \;\; S_ue_u]$.

Similarly, using the procedure for Eq. (5.16) and solving, we get

$$[w_2 \;\; b_2]^T = (W^TW + c_2V^TV + c_4I + c_uZ^TZ)^{-1}(c_2V^TS_1e_1 + c_uZ^TS_u(1-\epsilon)e_u). \tag{5.21}$$

A new data point x is classified using the decision function in Eq. (2.11).

5.2.2 Non-linear FULSTSVM

The formulation of non-linear FULSTSVM is written as

$$\begin{aligned}\min_{w_1,b_1,\xi_1,\psi_1} \quad & \frac{1}{2}\|S_1(K(X_1,D^T)w_1 + e_1b_1)\|^2 + \frac{c_1}{2}\|S_2\xi_1\|^2 + \frac{c_3}{2}(\|w_1\|^2 + b_1^2) + \frac{c_u}{2}\|S_u\psi_1\|^2 \\ s.t. \quad & -(K(X_2,D^T)w_1 + e_2b_1) + \xi_1 = e_2, \\ & K(U,D^T)w_1 + e_ub_1 + \psi_1 = (-1+\epsilon)e_u, \end{aligned} \tag{5.22}$$

$$\begin{aligned}\min_{w_2,b_2,\xi_2,\psi_2} \quad & \frac{1}{2}\|S_2(K(X_2,D^T)w_2 + e_2b_2)\|^2 + \frac{c_2}{2}\|S_1\xi_2\|^2 + \frac{c_4}{2}(\|w_2\|^2 + b_2^2) + \frac{c_u}{2}\|S_u\psi_2\|^2 \\ s.t. \quad & K(X_1,D^T)w_2 + e_1b_2 + \xi_2 = e_1, \\ & -(K(U,D^T)w_2 + e_ub_2) + \psi_2 = (-1+\epsilon)e_u, \end{aligned} \tag{5.23}$$

where $K(X_i, D^T)$ is the kernel matrix, $D = [X_1; X_2]$, S_i, S_u are diagonal fuzzy membership matrices of data points in the classes and universum respectively, $i = 1, 2$.

Rewriting the objective functions using the constraints, we get

$$\begin{aligned}\min_{w_1,b_1} \quad & \frac{1}{2}\|S_1(K(X_1,D^T)w_1 + e_1b_1)\|^2 + \frac{c_1}{2}\|S_2(K(X_2,D^T)w_1 + e_2b_1 + e_2)\|^2 \\ & + \frac{c_3}{2}(\|w_1\|^2 + b_1^2) + \frac{c_u}{2}\|S_u(-(K(U,D^T)w_1 + e_ub_1) + (-1+\epsilon)e_u)\|^2, \end{aligned} \tag{5.24}$$

$$\begin{aligned}\min_{w_2,b_2} \quad & \frac{1}{2}\|S_2(K(X_2,D^T)w_2 + e_2b_2)\|^2 + \frac{c_2}{2}\|S_1(-(K(X_1,D^T)w_2 + e_1b_2) + e_1)\|^2 \\ & + \frac{c_4}{2}(\|w_2\|^2 + b_2^2) + \frac{c_u}{2}\|S_u(K(U,D^T)w_2 + e_ub_2 + (-1+\epsilon)e_u)\|^2. \end{aligned} \tag{5.25}$$

The parameters w_1 and b_1 are obtained by setting the gradient of QPP (5.24) w.r.t. w_1 and b_1 equal to 0, and solving we get,

$$[w_1 \;\; b_1]^T = -\left(M^T M + c_1 N^T N + c_3 I + c_u O^T O\right)^{-1}\left(c_1 N^T S_2 e_2 + c_u O^T S_u (1-\epsilon) e_u\right), \tag{5.26}$$

where $M = [S_1 K(X_1, D^T) \quad S_1 e_1]$, $N = [S_2 K(X_2, D^T) \quad S_2 e_2]$, and $O = [S_u K(U, D^T) \;\; S_u e_u]$. Similarly, using Eq. (5.25), we get

$$[w_2 \;\; b_2]^T = \left(N^T N + c_2 M^T M + c_4 I + c_u O^T O\right)^{-1}\left(c_2 M^T S_1 e_1 + c_u O^T S_u (1-\epsilon) e_u\right). \tag{5.27}$$

For a new data point, similar to linear case, the class is assigned based on the class of the nearest hyperplane using Eq. (2.10). In the following subsection, we present the fuzzy membership function used in the proposed FULSTSVM.

5.2.3 Fuzzy membership function

The proposed FULSTSVM utilizes a fuzzy function inspired by [41]. The following fuzzy function keeps the range of fuzzy memberships in the range (0.5, 1]. The membership function is described as

$$f(x_i) = 1 - 0.5\left(\frac{|x_i - c_j|}{r_j + \rho}\right), \tag{5.28}$$

where x_i is a data point belonging to class j with centre c_j, $i = 1, \ldots, m_j, j = 1, 2$. The variable r_j is the largest distance from the class centre of data points of class j, and ρ is a very small positive value to avoid division by zero.

The range of fuzzy memberships in the above mentioned fuzzy function is chosen as (0.5, 1]. This is to keep significant contribution of majority of the data points in the formation of the classifier. Moreover, the proposed FULSTSVM also gives fuzzy memberships to the universum data points. The contribution of most universum data points is required for providing prior information about the data, which is achieved

by this function. Moreover, the contribution of outliers is reduced accordingly. This approach is in contrast to the approach proposed in FSVM [41], where the fuzzy memberships are chosen in the range (0,1].

5.2.4 Time complexity

The time complexity of TWSVM is $2 * O(m/2)^3$ i.e., $O(m)^3/4$, where m is total number of data points [12]. This is the time complexity of solving the QPPs, which is a computationally intensive task. Moreover, TWSVM involves two matrix inverses having a complexity of $O(n)^3$, where n is the dimension of the matrix [167]. Similarly, UTWSVM has time complexity of $O(m + 2u)^3/4$, where u denotes the number of universum data points.

On the other hand, the formulation of LSTSVM involves solution of linear equations using two matrix inverses. Therefore, the computation time of LSTSVM is lesser than TWSVM and UTSVM in Table 5.3. Similarly, ULSTSVM involves two inverses with additional universum data. The time complexity of proposed FULSTSVM is similar to ULSTSVM, with additional complexity for fuzzy membership function. The complexity of fuzzy membership is $O(m)$. Hence, the computation of FULSTSVM is more than ULSTSVM, but the additional time is $O(m)$, which is insignificant w.r.t. cubic complexity of inverse calculation in FULSTSVM and ULSTSVM.

5.2.5 Experimental results

In this section, we perform numerical experiments, and show the comparative analysis of the results obtained on benchmark datasets. We also present two biomedical applications viz. Alzheimer's disease and breast cancer to show the utility of the proposed FULSTSVM. The experiments are performed on a PC running on 64 bit Windows 10 operating system, with 2.30 GHz Intel® Xeon processor, and 128 GB of RAM with MATLAB R2017a environment.

5.2.5.1 Parameter settings

For experiments on real world datasets, the parameters are selected as follows: Penalty parameters are set as $c_1 = c_2 = c_u$, and $c_3 = c_4$. The range for the penalty parameters, and μ is same as in subsection 3.1.5. The parameter ϵ is selected from $\{0.2, 0.4, 0.6, 0.8\}$. The universum is generated by averaging the samples randomly from the data [20,27]. The training and testing data is chosen as 50% of total samples. For large scale datasets, we use fixed value of the hyper-parameters [31,158]. Therefore, the value of $c_1 = c_2 = c_u$ is fixed as 10, and $c_3 = c_4$ is set as 10^{-5}, ϵ is selected as 0.7, and μ is chosen as 2 for all the algorithms.

In biomedical datasets, we used 150 structural MRI (T1) images from the ADNI database. The pre-processing and other specifications of the MRI images are same as in 4.2.2. For breast cancer, the BreakHis histopathological dataset is utilized [223] in this work. A total of 314 histopathological breast tissue images include a benign condition i.e., adenosis (ADN), and a cancer i.e., ductal carcinoma (DC). The histopathological images are converted to gray level, and features are extracted using wavelet transform (Daubechies-4) up to 3 levels of decomposition. The approximation and detail coefficients are concatenated to form the feature vector [224].

5.2.5.2 Real world data

The results on 18 real world benchmark datasets are presented in Table 5.3. For comparison, we used TWSVM [12], UTSVM [27], LSTSVM [18], and ULSTSVM [71] algorithms. One can observe in Table 5.3 that the proposed FULSTSVM obtained the highest accuracies in 11 datasets. FULSTSVM outperformed the existing algorithms by obtaining an average rank of 1.8056 on accuracy values. This is due to the use of fuzzy memberships for all data points in the proposed FULSTSVM. It is noticeable that the proposed FULSTSVM achieved highest accuracy of 98.54% for Breast cancer wisconsion dataset with LSTSVM. However, ULSTSVM achieved a lesser accuracy of 98.25%, due to equal weighting to all universum data points in ULSTSVM.

One can observe in Table 5.3 that the training time of proposed FULSTSVM is

Table 5.3: Comparative performance of proposed algorithm with existing approaches for classification on real world benchmark datasets. Accuracy is in percentage, and average rank is calculated on accuracy.

Dataset (Size)	**TWSVM** [12] Accuracy (c_1, μ) Time (s)	**UTSVM** [27] Accuracy (c_1, μ, ϵ) Time (s)	**LSTSVM** [18] Accuracy (c_1, μ) Time (s)	**ULSTSVM** [71] Accuracy $(c_1, c_3, \epsilon, \mu)$ Time (s)	**Proposed FULSTSVM** Accuracy $(c_1, c_3, \epsilon, \mu)$ Time (s)
Ecoli-0-1_vs_5 (240×12)	93.39 $(10^{-5}, 2^5)$ 0.2566	**95.04** $(10^{-4}, 2^4, 0.6)$ 0.2598	93.39 $(10^{-5}, 2^5)$ 0.0221	**95.04** $(10^{-5}, 10^{-1}, 0.2, 2^4)$ 0.0261	**95.04** $(10^{-5}, 10^{-2}, 0.2, 2^4)$ 0.0388
Ecoli-0-1-4-7_vs_5-6 (332×12)	95.81 $(10^{-4}, 2^5)$ 0.0794	98.2 $(10^{-3}, 2^4, 0.4)$ 0.0781	97.6 $(10^{-1}, 2^5)$ 0.0321	**98.8** $(10^{0}, 10^{-1}, 0.6, 2^5)$ 0.0364	**98.8** $(10^{0}, 10^{-2}, 0.6, 2^5)$ 0.0444
Ecoli-0-2-3-4_vs_5 (202×7)	98.04 $(10^{-3}, 2^4)$ 0.0881	98.04 $(10^{-4}, 2^5, 0.4)$ 0.0898	97.06 $(10^{1}, 2^5)$ 0.0118	94.12 $(10^{-2}, 10^{-5}, 0.4, 2^4)$ 0.0129	**99.02** $(10^{-2}, 10^{-5}, 0.6, 2^5)$ 0.0161
Ecoli-0-2-6-7_vs_3-5 (224×7)	93.81 $(10^{-2}, 2^5)$ 0.0332	94.69 $(10^{-2}, 2^5, 0.4)$ 0.0379	**97.35** $(10^{2}, 2^5)$ 0.0141	92.92 $(10^{-1}, 10^{-5}, 0.6, 2^5)$ 0.0157	94.69 $(10^{0}, 10^{-4}, 0.6, 2^5)$ 0.019
Ecoli-0-4-6_vs_5 (202×6)	94.12 $(10^{-5}, 2^5)$ 0.0274	94.12 $(10^{-4}, 2^5, 0.8)$ 0.0384	94.12 $(10^{3}, 2^5)$ 0.0155	95.1 $(10^{0}, 10^{-3}, 0.6, 2^5)$ 0.018	**96.08** $(10^{0}, 10^{-5}, 0.4, 2^5)$ 0.0189
Ecoli-0-6-7_vs_3-5 (222×7)	91.07 $(10^{-4}, 2^4)$ 0.0295	91.07 $(10^{-3}, 2^4, 0.4)$ 0.0333	**94.64** $(10^{1}, 2^5)$ 0.0137	93.75 $(10^{-1}, 10^{-5}, 0.4, 2^5)$ 0.0156	93.75 $(10^{1}, 10^{-3}, 0.2, 2^5)$ 0.0177
Glass4 (214×9)	94.44 $(10^{-5}, 2^0)$ 0.0341	95.37 $(10^{-1}, 2^3, 0.2)$ 0.0304	94.44 $(10^{-5}, 2^0)$ 0.0127	95.37 $(10^{-1}, 10^{-5}, 0.6, 2^1)$ 0.0108	**97.22** $(10^{0}, 10^{-4}, 0.6, 2^1)$ 0.0192
Vehicle 1 (846×18)	**74.06** $(10^{-4}, 2^5)$ 0.2701	73.82 $(10^{-4}, 2^5, 0.8)$ 0.3207	73.82 $(10^{-4}, 2^5)$ 0.2116	72.88 $(10^{-4}, 10^{-5}, 0.6, 2^5)$ 0.2364	73.82 $(10^{-4}, 10^{-5}, 0.2, 2^5)$ 0.2319
Vehicle2 (846×18)	96.7 $(10^{-3}, 2^5)$ 0.2824	**98.11** $(10^{-3}, 2^5, 0.6)$ 0.243	**98.11** $(10^{-2}, 2^5)$ 0.1469	**98.11** $(10^{0}, 10^{-3}, 0.6, 2^5)$ 0.2063	**98.11** $(10^{-1}, 10^{-5}, 0.6, 2^5)$ 0.2338
Pima Indians (768×8)	75.84 $(10^{-5}, 2^5)$ 0.1783	76.1 $(10^{-5}, 2^5, 0.8)$ 0.1807	74.81 $(10^{-5}, 2^5)$ 0.1164	74.29 $(10^{-5}, 10^{-5}, 0.2, 2^5)$ 0.1264	**78.96** $(10^{1}, 10^{1}, 0.6, 2^5)$ 0.1369
Yeast3 (1484×8)	**94.35** $(10^{-1}, 2^0)$ 0.6316	93.41 $(10^{3}, 2^2, 0.6)$ 0.7756	93.67 $(10^{-2}, 2^0)$ 0.463	93.81 $(10^{-2}, 10^{-5}, 0.2, 2^0)$ 0.5962	94.21 $(10^{3}, 10^{3}, 0.8, 2^{-2})$ 0.495
Yeast1vs7 (458×8)	92.61 $(10^{-1}, 2^{-1})$ 0.0933	93.04 $(10^{-1}, 2^0, 0.4)$ 0.079	92.61 $(10^{0}, 2^{-1})$ 0.0434	**93.48** $(10^{-2}, 10^{-3}, 0.4, 2^{-2})$ 0.0471	**93.48** $(10^{-2}, 10^{-3}, 0.4, 2^{-2})$ 0.0593

Table 5.3 (contd.)

Dataset (Size)	**TWSVM** [12] Accuracy (c_1, μ) Time (s)	**UTSVM** [27] Accuracy (c_1, μ, ϵ) Time (s)	**LSTSVM** [18] Accuracy (c_1, μ) Time (s)	**ULSTSVM** [71] Accuracy $(c_1, c_3, \epsilon, \mu)$ Time (s)	**Proposed FULSTSVM** Accuracy $(c_1, c_3, \epsilon, \mu)$ Time (s)
Ecoli0137vs26 (311×7)	**97.44** $(10^{-1}, 2^3)$ 0.0364	96.15 $(10^0, 2^0, 0.4)$ 0.0428	94.87 $(10^{-3}, 2^{-2})$ 0.01876	95.51 $(10^{-2}, 10^{-1}, 0.2, 2^1)$ 0.0206	96.15 $(10^{-5}, 10^0, 0.2, 2^{-1})$ 0.0244
Yeast5 (1484×8)	96.64 $(10^0, 2^4)$ 0.805	96.64 $(10^{-4}, 2^{-2}, 0.2)$ 0.7354	96.5 $(10^{-5}, 2^{-1})$ 0.4878	96.5 $(10^{-5}, 10^{-5}, 0.4, 2^{-1})$ 0.5705	**96.77** $(10^{-4}, 10^{-3}, 0.6, 2^{-2})$ 0.5929
Cleveland (297×13)	81.21 $(10^{-1}, 2^4)$ 0.0289	76.51 $(10^{-2}, 2^3, 0.8)$ 0.0319	81.21 $(10^1, 2^4)$ 0.0172	81.88 $(10^2, 10^3, 0.6, 2^2)$ 0.0194	**83.22** $(10^{-1}, 10^{-4}, 0.6, 2^5)$ 0.0254
Transfusion (748×4)	78.93 $(10^{-3}, 2^5)$ 0.2164	78.93 $(10^{-3}, 2^5, 0.2)$ 0.2377	81.07 $(10^{-3}, 2^5)$ 0.1115	**82.67** $(10^2, 10^1, 0.4, 2^5)$ 0.1207	81.87 $(10^1, 10^1, 0.4, 2^5)$ 0.1559
Breast cancer wisconsin (682×9)	98.25 $(10^{-4}, 2^3)$ 0.1431	98.25 $(10^{-4}, 2^3, 0.2)$ 0.1971	**98.54** $(10^0, 2^5)$ 0.0855	98.25 $(10^0, 10^1, 0.8, 2^4)$ 0.095	**98.54** $(10^3, 10^2, 0.4, 2^5)$ 0.1408
Ripley (1250×2)	90.58 $(10^0, 2^{-1})$ 0.5155	91.05 $(10^0, 2^{-1}, 0.4)$ 0.5511	**91.21** $(10^0, 2^{-1})$ 0.3014	91.05 $(10^3, 10^5, 0.8, 2^{-3})$ 0.3378	91.05 $(10^{-2}, 10^{-5}, 0.4, 2^{-1})$ 0.4951
Average accuracy	90.96	91.03	91.39	91.31	**92.27**
Average rank	3.5556	3.2222	3.3056	3.1111	**1.8056**

lower than TWSVM and UTSVM algorithms. This is because TWSVM and UTSVM solve a pair of QPPs, which is computationally expensive. On the other hand proposed FULSTSVM solves a system of linear equations. However, the training time of FULSTSVM is higher than LSTSVM and ULSTSVM due to the fuzzy memberships.

5.2.5.3 Statistical significance

In order to prove the statistical significance of the proposed FULSTSVM for generalization performance, we perform the Friedman and Nemenyi posthoc test [172].

The Friedman test is performed using the average ranks of the algorithms from Table 5.3. Here, we first assume that all the algorithms are not significantly different, as the null hypothesis. We calculated the χ_F^2 value as 13.6151.

The F_F value is obtained as

$$F_F = \frac{(18-1)(13.6151)}{18 \times (5-1) - 13.6151} = 3.9643.$$

In this case, the F-distribution has $\big(5-1, (5-1)(18-1)\big) = (4, 68)$ degrees of freedom. Therefore, the critical value for $F(6, 150)$ at $\alpha = 0.05$ level of significance is 2.5066. Since $F_F = 3.9643 > 2.5066$, we reject the null hypothesis. Thus, there is significant difference between these methods.

Next, for pairwise difference, we use the Nemenyi posthoc test [172] to check pairwise difference between proposed FULSTSVM and existing algorithms. The critical difference (CD) for our case at $\alpha = 0.10$ level of significance is $2.459\sqrt{\frac{5(5+1)}{6\times 18}} = 1.296$. The pairwise difference of the average ranks should be greater than CD for significance. Table 5.4 shows the pairwise significant difference between the methods based on average ranks. One can observe that proposed FULSTSVM is significantly different from TWSVM, UTSVM, LSTSVM, and ULSTSVM algorithms.

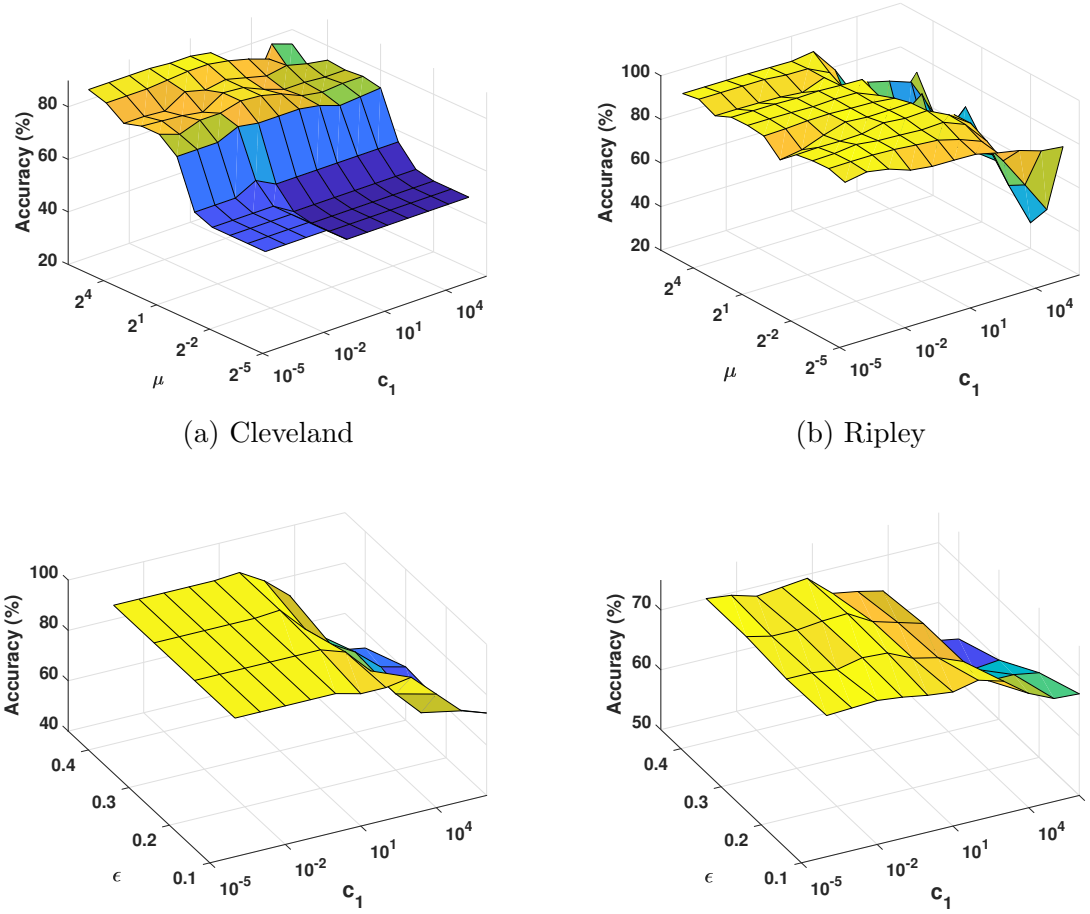
Table 5.4: Significant difference between the proposed FULSTSVM and existing algorithms in pairwise comparison.

Significance	**TWSVM**	**UTSVM**	**LSTSVM**	**ULSTSVM**
Proposed FULSTSVM	**Yes**	**Yes**	**Yes**	**Yes**

5.2.5.4 Insensitivity analysis

To check the effect of hyper-parameter values on the accuracy of the proposed FULSTSVM, we present the insensitive analysis. The insensitivity performance of FULSTSVM is shown for varying values of c_1, μ, and ϵ hyper-parameters in Fig. 5.3.

Figs. 5.3(a) and 5.3(b) show the change in accuracy for different values of c_1 and μ. One can observe that accuracy of proposed FULSTSVM is higher for lower values of c_1, and higher values of μ. The variation in accuracy for c_1 with ϵ is shown in Figs. 5.3(c) and 5.3(d). The parameter ϵ is not affecting the accuracy in a significant manner. However, here also the accuracy of FULSTSVM is higher for lesser values of the hyper-parameter c_1. This also justifies the parameter selection in the experiments.



(a) Cleveland

(b) Ripley

(c) Ecoli-0-1_vs_5

(d) Vehicle1

Figure 5.3: Insensitivity analysis of proposed FULSTSVM for c_1 and μ in (a) and (b), and for c_1 and ϵ in (c) and (d) on real world benchmark datasets.

5.2.5.5 Biomedical data

In this section, we present the results on classification of Alzheimer's disease and breast cancer datasets. The results for these applications are shown in Table 5.5. One can observe that the proposed FULSTSVM performs better than baseline algorithms in most of the cases. This is reflected in the average rank based on accuracy. The proposed FULSTSVM obtained lowest average rank of 2.5. The accuracy of FULSTSVM is higher than other algorithms for classification of AD vs MCI shown in AD_MCI, which is a difficult classification problem [2].

Moreover, for breast cancer data i.e. adenosis vs ductal carcinoma, the proposed FULSTSVM obtains highest accuracy of 84.18%. The better average rank of proposed

Table 5.5: Comparative performance of the proposed and baseline algorithms on Alzheimer's and breast cancer datasets. Accuracy is in percentage.

Dataset	**TWSVM** [12] Accuracy (c_1, μ) Time (s)	**UTSVM** [27] Accuracy (c_1, μ, ϵ) Time (s)	**LSTSVM** [18] Accuracy (c_1, μ) Time (s)	**ULSTSVM** [71] Accuracy $(c_1, c_3, \epsilon, \mu)$ Time (s)	**Proposed FULSTSVM** Accuracy $(c_1, c_3, \epsilon, \mu)$ Time (s)
CN_AD	**80** $(10^{-1}, 2^5)$ 0.255	**80** $(10^{-1}, 2^5, 0.2)$ 0.2427	77.5 $(10^{-5}, 2^1)$ 0.0105	75 $(10^1, 10^4, 0.8, 2^4)$ 0.0149	72.5 $(10^{-5}, 10^4, 0.2, 2^4)$ 0.0255
CN_MCI	69.23 $(10^{-5}, 2^5)$ 0.0311	66.67 $(10^{-2}, 2^2, 0.8)$ 0.0328	69.23 $(10^{-5}, 2^5)$ 0.0062	**71.79** $(10^{-1}, 10^{-4}, 0.2, 2^5)$ 0.0081	69.23 $(10^0, 10^{-3}, 0.2, 2^4)$ 0.0109
AD_MCI	53.85 $(10^{-5}, 2^5)$ 0.094	58.97 $(10^{-1}, 2^0, 0.2)$ 0.0677	48.72 $(10^{-1}, 2^5)$ 0.006	48.72 $(10^{-2}, 10^{-5}, 0.8, 2^{-1})$ 0.0054	**66.67** $(10^{-5}, 10^{-4}, 0.6, 2^5)$ 0.0094
ADN_DC	83.54 $(10^{-5}, 2^3)$ 0.2538	83.54 $(10^{-4}, 2^4, 0.4)$ 0.2968	83.54 $(10^{-5}, 2^3)$ 0.0732	82.28 $(10^{-5}, 10^0, 0.2, 2^4)$ 0.0939	**84.18** $(10^{-5}, 10^{-1}, 0.2, 2^5)$ 0.1072
Average rank	2.625	2.875	3.375	3.625	**2.5**

FULSTSVM for accuracy can be attributed to the use of fuzzy membership with universum data. It leads to prior information for the model, with less sensitivity to outlier data points of the classes, as well as the universum. This implies the applicability of the proposed FULSTSVM for biomedical applications.

5.2.5.6 Large scale data

In order to check the performance of the proposed FULSTSVM on large datasets, we used the Skin segmentation dataset from UCI repository [170]. For comparison, we used two other efficient algorithms viz. LSTSVM and ULSTSVM. The results are shown in Table 5.6. It is observable that the proposed FULSTSVM is showing higher accuracy on most of the datasets. This is because FULSTSVM removes the effect of outliers in the generation of universum data, whereas ULSTSVM gives equal importance to all the universum data points. Moreover, the proposed FULSTSVM also gives proper weighting to the data points of the binary classes, where LSTSVM

Table 5.6: Classification performance of the proposed and baseline algorithms on large scale datasets. Accuracy is in percentage.

Dataset	**LSTSVM** [18] Accuracy Time (s)	**ULSTSVM** [71] Accuracy Time (s)	**Proposed FULSTSVM** Accuracy Time (s)
Skin-5k	93.56 4.9931	**97.8** 5.7513	97.76 6.1476
Skin-10k	95.26 20.8591	97.64 24.0887	**97.88** 24.4975
Skin-20k	97.03 97.2516	98.8 104.539	**98.91** 105.941
Skin-30k	97.74 233.092	99.23 243.475	**99.25** 247.076
Skin-40k	98.6 427.891	99.47 453.43	**99.53** 548.742

gives equal to weights to all the data points. However, the time is the least in case of LSTSVM, because there is no universum data in LSTSVM. The time is slightly higher in FULSTSVM as compared to ULSTSVM due to the calculation of fuzzy membership values. However, the additional time in FULSTSVM is not significant as discussed in section 5.2.4 in terms of time complexity.

5.3 Summary

In this chapter, for dealing with noisy data in universum learning, we have proposed three novel fuzzy based SVM algorithms. We proposed a fuzzy based approach for USVM and UTSVM algorithms which is useful in the classification of data with noise and outliers. The proposed FUSVM and FUTSVM have shown better generalization performance for most of the datasets. This fuzzy based approach for universum helps in giving prior information to the data in an effective manner. The use of information entropy of the universum points is helpful in giving optimum membership values to the universum data points.

Moreover, we proposed a more efficient fuzzy based learning algorithm, termed as fuzzy universum least squares twin support vector machine (FULSTSVM). The proposed algorithm gives prior information about data distribution to the classifier, and also provides fuzzy membership to the data points and universum. Proposed FULSTSVM also performed better on large sized datasets in terms of accuracy, showing its scalability on large datasets. Results on applications i.e. Alzheimer's disease and breast cancer clearly show the applicability of the proposed FULSTSVM for healthcare data.

However, the calculation of fuzzy memberships involved in the works of this chapter incur additional computation time. Moreover, in universum based algorithms, there is a drawback of higher training time due to the additional universum data. To address the issue of computation complexity in universum learning, we present efficient universum based SVM algorithms for classification problems in the next chapter.

Chapter 6

Efficient universum twin support vector machines

In this chapter, we present some novel formulations for improving the efficiency of universum based SVM algorithms. Each of these proposed formulations introduce some important ideas to improve the performance of universum based algorithms on classification tasks. Section 6.1 presents a novel angle based approach for universum learning termed as angle based universum least squares twin support vector machine (AULSTSVM)[1]. In section 6.2, we present an efficient least squares based algorithm with universum data, known as universum least squares twin parametric-margin SVM (ULSTPMSVM)[2]. Lastly, section 6.3 presents an improved version of universum twin SVM using regularization, abbreviated as IUTSVM[3].

[1]**B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. An efficient angle based universum least squares twin support vector machine for pattern classification. *ACM Transactions on Internet Technology (TOIT)*, ACM, 2021, DOI: `https://doi.org/10.1145/3387131`.
[**SCI Indexed Impact Factor: 3.135**]

[2]**B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. Universum least squares twin parametric-margin support vector machine. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1-8. IEEE, 2020, DOI: `https://doi.org/10.1109/IJCNN48605.2020.9206865`.
[**Scopus Indexed, Core rank: A**]

[3]**B. Richhariya**, A. Sharma, M. Tanveer. Improved universum twin support vector machine. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2045-2052. IEEE, 2018, DOI: `https://doi.org/10.1109/SSCI.2018.8628671`.
[**Scopus Indexed**]

6.1 An efficient angle based universum least squares twin support vector machine for pattern classification

Universum based support vector machine (USVM) incorporates prior information about the distribution of data in training of the classifier. This leads to better generalization performance, but with increased computation cost. Various twin hyperplane based models are proposed to reduce the computation cost of universum based algorithms. Khemchandani et al. [225] proposed an angle based twin support vector machine (ATWSVM), and angle based least squares twin support vector machine (LS-ATWSVM). Motivated by this approach, we present an efficient angle based universum least squares twin support vector machine (AULSTSVM) for classification. This is a novel approach of incorporating universum in the formulation of least squares based twin SVM model.

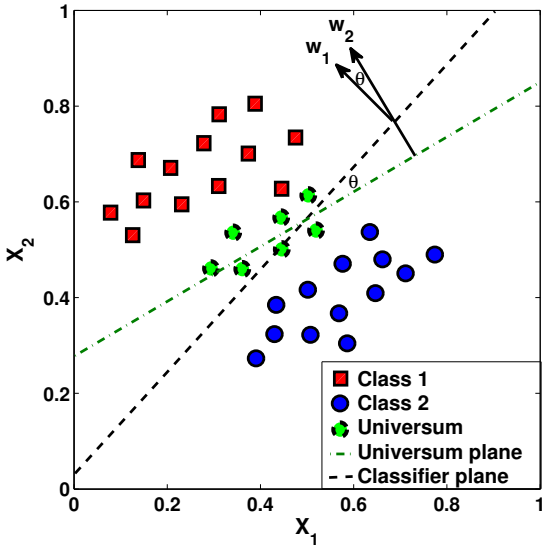


Figure 6.1: Classification of data points by the proposed AULSTSVM.

The geometrical representation of the proposed approach is shown in Fig. 6.1. In contrast to TWSVM and LSTSVM where twin hyperplanes are proximal to the binary classes, in proposed AULSTSVM one hyperplane remains at minimum distance from

the universum data points, while other hyperplane remains at minimum distance from the data samples of the binary classes. The objective of the proposed AULSTSVM is to minimize the angle (θ) between the two hyperplanes. By this approach, the universum plane provides prior information about the data distribution to the classifier.

In contrast to ATWSVM and LS-ATWSVM, where the angle between the twin hyperplanes is maximized, AULSTSVM minimizes the angle between the twin hyperplanes. This is due to introduction of the idea of universum hyperplane in AULSTSVM. In the proposed approach, the angle is minimized by maximizing the dot product of the normal vectors, i.e., $w_1^T w_2$. The following subsections present the formulations of the proposed AULSTSVM for the linear and non-linear cases.

6.1.1 Linear AULSTSVM

There are two optimization problems in AULSTSVM, where the first QPP (6.1) constructs a hyperplane proximal to universum data, while the second QPP (6.2) constructs the classifier. A regularization term i.e., $\frac{c_1}{2}(\|w_1\|^2 + b_1^2)$ is added to control the model complexity, as well as to remove the ill-conditioning of the matrices [33].

The optimization problems of linear AULSTSVM are written as

$$\begin{aligned} \min_{w_1,b_1,\xi_1,\eta_1} \quad & \frac{c_1}{2}(\|w_1\|^2 + b_1^2) + \frac{1}{2}\|Uw_1 + e_u b_1\|^2 + c_3 e_1^T \xi_1 + c_5 e_2^T \eta_1 \\ s.t. \quad & -(X_1 w_1 + e_1 b_1) = \xi_1, \\ & X_2 w_1 + e_2 b_1 = \eta_1, \end{aligned} \tag{6.1}$$

$$\begin{aligned} \min_{w_2,b_2,\xi_2,\eta_2} \quad & \frac{c_2}{2}(\|w_2\|^2 + b_2^2) + \frac{c_4}{2}\|\xi_2\|^2 + \frac{c_6}{2}\|\eta_2\|^2 - c_7(w_1^T w_2 + b_1 b_2) \\ s.t. \quad & X_1 w_2 + e_1 b_2 - e_1 = \xi_2, \\ & X_2 w_2 + e_2 b_2 + e_2 = \eta_2, \end{aligned} \tag{6.2}$$

where $c_i > 0,\ i = 1, \ldots 7$ are positive parameters, $w_i,\ i = 1, 2$ represent the weight vectors, and $\xi_i, \eta_i,\ i = 1, 2$ represent the slack variables.

The objective function in QPP (6.1) minimizes the distance of universum data points from the universum hyperlane, while keeping the data points of the binary classes as close as possible. QPP (6.1) solves the optimization problem using the linear loss function, making the classifier less sensitive towards outliers. Consequently, the computation cost of the proposed AULSTSVM is reduced in comparison to algorithms like ULSTSVM, where quadratic loss is used for all the data points (Eqs. 2.30 and 2.31). Moreover, in QPP (6.2), the objective function minimizes the distance of classifying hyperplane from the binary classes, while minimizing the angle between the two hyperplanes. This is a novel formulation for least squares based SVM models by including information about data distribution using an angle based approach.

The unconstrained optimization problem (UOP) for Eq. (6.1) is written as

$$L_1 = \frac{c_1}{2}(\|w_1\|^2 + b_1^2) + \frac{1}{2}\|Uw_1 + e_u b_1\|^2 - c_3 e_1^T(X_1 w_1 + e_1 b_1) + c_5 e_2^T(X_2 w_1 + e_2 b_1). \tag{6.3}$$

Similarly, the UOP for Eq. (6.2) is given as

$$\begin{aligned} L_2 = &\frac{c_2}{2}(\|w_2\|^2 + b_2^2) + \frac{c_4}{2}\|X_1 w_2 + e_1 b_2 - e_1\|^2 + \frac{c_6}{2}\|X_2 w_2 + e_2 b_2 + e_2\|^2 \\ &- c_7(w_1^T w_2 + b_1 b_2). \end{aligned} \tag{6.4}$$

Taking the gradient of L_1 (Eq. 6.3) w.r.t. w_1 and b_1 and equating to 0, we get

$$c_1 w_1 + U^T(Uw_1 + e_u b_1) - c_3 X_1^T e_1 + c_5 X_2^T e_2 = 0, \tag{6.5}$$

$$c_1 b_1 + e_u^T(Uw_1 + e_u b_1) - c_3 e_1^T e_1 + c_5 e_2^T e_2 = 0. \tag{6.6}$$

Rearranging the terms in Eqs. (6.5) and (6.6), and solving for w_1 and b_1, we get

$$\begin{aligned} &c_1 u_1 + O^T O u_1 - c_3 H^T e_1 + c_5 G^T e_2 = 0, \\ &u_1 = (O^T O + c_1 I)^{-1}(c_3 H^T e_1 - c_5 G^T e_2), \end{aligned} \tag{6.7}$$

where $H = [X_1 \;\; e_1]$, $G = [X_2 \;\; e_2]$, $O = [U \;\; e_u]$, and $u_1 = [w_1 \;\; b_1]^T$. Similarly, setting

the gradient of L_2 (Eq. 6.4) w.r.t. w_2 and b_2 equal to 0, we get

$$c_2 w_2 + c_4 X_1^T (X_1 w_2 + e_1 b_2 - e_1) + c_6 X_2^T (X_2 w_2 + e_2 b_2 + e_2) - c_7 w_1 = 0, \tag{6.8}$$

$$c_2 b_2 + c_4 e_1^T (X_1 w_2 + e_1 b_2 - e_1) + c_6 e_2^T (X_2 w_2 + e_2 b_2 + e_2) - c_7 b_1 = 0. \tag{6.9}$$

Rearranging the terms in Eqs. (6.8) & (6.9) and solving for $u_2 = [w_2 \;\; b_2]^T$, we get

$$u_2 = (c_4 H^T H + c_6 G^T G + c_2 I)^{-1} (c_4 H^T e_1 - c_6 G^T e_2 + c_7 u_1). \tag{6.10}$$

The decision function for a new data point x is given as

$$f(x) = sgn(w_2^T x + b_2). \tag{6.11}$$

Note: The decision function of the proposed AULSTSVM (Eq. 6.11) is different than that of UTSVM, LSTSVM and ULSTSVM (Eq. 2.11).

The algorithm for linear AULSTSVM is described in the following Alg. 6.1.

Algorithm 6.1 Linear AULSTSVM

1: ***Inputs:***
   1.1 Training samples $X \in \mathbb{R}^n$, training labels $Y \in \{1, -1\}$.
2: ***Generate universum data:***
   2.1 Select samples randomly from binary classes in matrices A and B.
   2.2 Compute the averages of samples as $U = (A + B)/2$.
3: ***Select optimal parameters:***
   3.1 Obtain optimal values for parameters i.e., $c_i > 0,\ i = 1, \ldots 7$ using k-fold cross validation on training data.
4: ***Construct plane for universum:***
   4.1 Generate plane for universum data i.e., $w_1^T x + b_1$ using Eq. (6.7).
5: ***Construct classifier:***
   5.1 Generate classifying plane i.e., $w_2^T x + b_2$ using Eq. (6.10).
6: ***Classification of testing data:***
   6.1 Assign labels to the data points based on sign of decision function in Eq. (6.11).

6.1.2 Non-linear AULSTSVM

The optimization problems of non-linear AULSTSVM are described as

$$\begin{aligned}\min_{w_1,b_1,\xi_1,\eta_1} \quad & \frac{c_1}{2}(\|w_1\|^2+b_1^2)+\frac{1}{2}\|K(U,D^T)w_1+e_ub_1\|^2+c_3e_1^T\xi_1+c_5e_2^T\eta_1\\ s.t. \quad & -(K(X_1,D^T)w_1+e_1b_1)=\xi_1,\\ & K(X_2,D^T)w_1+e_2b_1=\eta_1,\end{aligned} \tag{6.12}$$

$$\begin{aligned}\min_{w_2,b_2,\xi_2,\eta_2} \quad & \frac{c_2}{2}(\|w_2\|^2+b_2^2)+\frac{c_4}{2}\|\xi_2\|^2+\frac{c_6}{2}\|\eta_2\|^2-c_7(w_1^Tw_2+b_1b_2)\\ s.t. \quad & K(X_1,D^T)w_2+e_1b_2-e_1=\xi_2,\\ & K(X_2,D^T)w_2+e_2b_2+e_2=\eta_2,\end{aligned} \tag{6.13}$$

where $c_i>0,\ i=1,\ldots,7$ are parameters, $w_i,\ i=1,2$ is weight vector, $\xi_i,\eta_i,\ i=1,2$ are slack variables, and $K(X_i,D^T)$ is the kernel matrix, where $D=[A;B]$.

The UOP for Eq. (6.12) is written as

$$\begin{aligned}L_1=&\frac{c_1}{2}(\|w_1\|^2+b_1^2)+\frac{1}{2}\|K(U,D^T)w_1+e_ub_1\|^2-c_3e_1^T(K(X_1,D^T)w_1+e_1b_1)\\ &+c_5e_2^T(K(X_2,D^T)w_1+e_2b_1).\end{aligned} \tag{6.14}$$

Similarly, the UOP for Eq. (6.13) is written as

$$\begin{aligned}L_2=&\frac{c_2}{2}(\|w_2\|^2+b_2^2)+\frac{c_4}{2}\|K(X_1,D^T)w_2+e_1b_2-e_1\|^2+\frac{c_6}{2}\|K(X_2,D^T)w_2+e_2b_2+e_2\|^2\\ &-c_7(w_1^Tw_2+b_1b_2).\end{aligned} \tag{6.15}$$

Setting the gradient of L_1 (Eq. 6.14) w.r.t. w_1 and b_1 and equating to 0, we get

$$c_1w_1+K(U,D^T)^T\big(K(U,D^T)w_1+e_ub_1\big)-c_3K(X_1,D^T)^Te_1+c_5K(X_2,D^T)^Te_2=0, \tag{6.16}$$

$$c_1b_1+e_u^T\big(K(U,D^T)w_1+e_ub_1\big)-c_3e_1^Te_1+c_5e_2^Te_2=0. \tag{6.17}$$

Rearranging the terms in Eqs. (6.16) and (6.17), and solving for w_1 and b_1, we get

$$\begin{aligned} c_1 u_1 + R^T R u_1 - c_3 P^T e_1 + c_5 Q^T e_2 = 0, \\ u_1 = (R^T R + c_1 I)^{-1}(c_3 P^T e_1 - c_5 Q^T e_2), \end{aligned} \tag{6.18}$$

where $P = [K(X_1, D^T) \;\; e_1]$, $Q = [K(X_2, D^T) \;\; e_2]$, $R = [K(U, D^T) \;\; e_u]$, and $u_1 = [w_1 \;\; b_1]^T$.

Similarly, we obtain $u_2 = [w_2 \;\; b_2]^T$ as

$$u_2 = (c_4 P^T P + c_6 Q^T Q + c_2 I)^{-1}(c_4 P^T e_1 - c_6 Q^T e_2 + c_7 u_1). \tag{6.19}$$

Here, Eqs. (6.18) and (6.19) involve two inverses of matrices having dimension $(m+1) \times (m+1)$, where $m = m_1 + m_2$. To reduce the computational complexity of inverse calculation, we use the Shermann-Morrison-Woodbury (SMW) formula [40] i.e.,

$$(A + BC^T)^{-1} = A^{-1} - A^{-1}B(I + C^T A^{-1} B)^{-1} C^T A^{-1}. \tag{6.20}$$

Using SMW approach in Eq. (6.18), we get

$$u_1 = \frac{1}{c_1}(I - R^T(c_1 I + RR^T)^{-1} R)(c_3 P^T e_1 - c_5 Q^T e_2). \tag{6.21}$$

In Eq. (6.19), we use the SMW approach twice and obtain two conditions. If $m_1 > m_2$, then we use the following:

$$V = \frac{1}{c_2}(I - c_4 P^T (c_2 I + c_4 P P^T)^{-1} P), \tag{6.22}$$

$$u_2 = \big(V^{-1} - c_6 V^{-1} Q^T (I + c_6 Q V^{-1} Q^T)^{-1} Q V^{-1}\big)\big(c_4 P^T e_1 - c_6 Q^T e_2 + c_7 u_1\big). \tag{6.23}$$

If $m_2 > m_1$, then u_2 is obtained as

$$W = \frac{1}{c_2}(I - c_6 Q^T (c_2 I + c_6 Q Q^T)^{-1} Q), \tag{6.24}$$

$$u_2 = \big(W^{-1} - c_4 W^{-1} P^T (I + c_4 P W^{-1} P^T)^{-1} P W^{-1}\big)\big(c_4 P^T e_1 - c_6 Q^T e_2 + c_7 u_1\big). \tag{6.25}$$

One can observe that in Eq. (6.21), one matrix inverse is calculated of size $m_3 \times m_3$. For u_2 (Eqs. (6.22-6.25), two matrix inverses need to be calculated of size $m_1 \times m_1$ and $m_2 \times m_2$. Hence, three inverses of smaller size are calculated using the SMW approach in comparison to two inverses of large size in Eqs. (6.18) and (6.19).

Finally, the decision function for a new data point x is given as

$$f(x) = sgn(K(x^T, D^T)w_2 + b_2). \tag{6.26}$$

The algorithm for non-linear AULSTSVM is given in the following Alg. 6.2.

Algorithm 6.2 Non-linear AULSTSVM

1: ***Inputs:***
   1.1 Training samples $X \in \mathbb{R}^n$, training labels $Y \in \{1, -1\}$.
2: ***Generate universum data:***
   2.1 Select samples randomly from binary classes in matrices A and B.
   2.2 Compute the averages of samples as $U = (A + B)/2$.
3: ***Select optimal parameters:***
   3.1 Obtain optimal values for parameters i.e., $c_i > 0,\ i = 1, \ldots 7$, and μ for the non-linear kernel function using k-fold cross validation on training data.
4: ***Construct kernel surface for universum:***
   4.1 Generate surface for universum data i.e., $K(x^T, D^T)w_1 + b_1$ using Eq. (6.21).
5: ***Construct non-linear classifier:***
   5.1 Generate the classifying surface i.e., $K(x^T, D^T)w_2 + b_2$ using the following
      if $(m_1 > m_2)$
      Use Eq. (6.23)
      else
      Use Eq. (6.25).
6: ***Classification of testing data:***
   6.1 Assign labels to the data points based on decision function in Eq. (6.26).

6.1.3 Proposed AULSTSVM vs ULSTSVM

For providing prior information about data distribution, the idea [10, 27] is to keep universum data within an ϵ-insensitive tube between the binary classes [27]. This is achieved by adding constraints for the universum which keeps the twin hyperplanes proximal to universum data. However, unlike UTSVM [27], the concept of ϵ-insensitive tube does not hold in case of ULSTSVM, due to the incorporation of quadratic loss instead of hinge loss. On the other hand, in case of proposed AULSTSVM, the prior information of universum is given to the classifier using an angle based approach. Consequently, the classifier of AULSTSVM aligns itself to the universum hyperplane, and also classifies the data points of the binary classes. This angle based approach was not used with universum data in the past. Moreover, the proposed AULSTSVM includes linear loss in one of the optimization problems, while ULSTSVM uses the quadratic loss function only. This leads to lesser computation cost of AULSTSVM as compared to ULSTSVM.

6.1.4 Proposed AULSTSVM vs ATWSVM and LS-ATWSVM

The formulation of ATWSVM solves a QPP and a system of linear equations to obtain the classifying hyperplane, whereas the proposed AULSTSVM only solves two systems of linear equations. In both ATWSVM and LS-ATWSVM, the angle between the two classifying hyperplane is maximized, whereas AULSTSVM minimizes the angle between the universum plane and the classifier.

Similar to the proposed AULSTSVM, LS-ATWSVM also solves two systems of linear equations. However, LS-ATWSVM uses the angle information in separating the data points of the binary classes [225]. On the contrary, the proposed AULSTSVM uses the angle information for aligning the classifier with the universum hyperplane. This results in prior information to the classifier for separating the data points of the binary classes.

6.1.5 Time complexity

The solution of proposed AULSTSVM involves computation of kernel matrices and matrix inverses. The time complexity for generation of kernel matrix is $O(m^2)$, while that of inverse calculation is $O(m^3)$ [167]. For addition of two square matrices, the complexity is $O(n^2)$, where n is the number of features in the dataset.

In linear case, the proposed AULSTSVM has lesser time complexity than ULSTSVM, due to the introduction of linear loss in place of quadratic. This leads to lesser number of matrix addition operations in the proposed AULSTSVM. The computation complexity of AULSTSVM is similar to LSTSVM and LS-ATWSVM. Moreover, the training time of proposed AULSTSVM is very less in comparison to TWSVM, ATWSVM, and UTSVM, as these algorithms involve solution of QPPs.

In non-linear case, SMW formula is used in the algorithms viz. LSTSVM, LS-ATWSVM, ULSTSVM, and proposed AULSTSVM. In this case also, AULSTSVM has lesser time complexity in comparison to ULSTSVM. This is due to lesser number of matrix additions, as well as lesser matrix inverses in AULSTSVM. The solution of AULSTSVM involves computation of 3 inverses, whereas ULSTSVM involves 5 inverses after applying the SMW approach. The other algorithms like LSTSVM, and LS-ATWSVM also involve the computation of 3 inverses. However, AULSTSVM and ULSTSVM involve the generation of kernel matrix for universum which has complexity of $O(m^2)$. This leads to additional computation time.

One important observation is that in comparison to LSTSVM and LS-ATWSVM, one matrix inverse in our proposed AULSTSVM is for universum, which is of very less size. A comparison of computation time of the algorithms is shown in the next subsection.

6.1.6 Experimental results

In this section, we present the comparison of proposed AULSTSVM with TWSVM [12], ATWSVM [225], UTSVM [27], LSTSVM [18], LS-ATWSVM [225], and ULSTSVM [71] algorithms. The experiments are performed for both linear and non-linear

cases of the algorithms. We used 26 benchmark datasets to analyze the performance of our proposed AULSTSVM. To show the applicability of the proposed AULSTSVM on real world problems, an application on Alzheimer's disease data is presented in this section. Moreover, an analysis of computation time is shown for the existing and proposed algorithms on large scale datasets. Statistical tests and insensitivity analysis are presented for detailed analysis of the proposed approach.

6.1.6.1 Parameter settings

In case of real world datasets in Table 6.1 and 6.2, 50% of data samples are used for training and rest for testing. The size of universum is chosen as 15% of total data samples in real world datasets. 5-fold cross-validation is used for selecting optimal parameters in all the algorithms. The parameters $c_i, i = 1, ..., 6$ are selected from the set $\{10^{-5}, 10^{-4}, ..., 10^{5}\}$ for TWSVM, ATWSVM, UTSVM, LSTSVM, and proposed AULSTSVM, while for ATWSVM and LS-ATWSVM parameters c_1, c_4 are chosen from $\{10^{-9}, 10^{-4}, ..., 10^{0}\}$ [225]. For UTSVM and ULSTSVM, ϵ is chosen from the set $\{0.1, 0.2, ..., 0.9\}$. To reduce the computational complexity of grid search [226], the parameters are set as $c_1 = c_2 = c_u$ in TWSVM and UTSVM, $c_1 = c_2, c_3 = c_4$ in ULSTSVM, and $c_1 = c_3 = c_5, c_2 = 1 - c_4$ [225] in ATWSVM and LS-ATWSVM. In proposed AULSTSVM the parameters are set as $c_1 = c_2, c_3 = c_5 = c_1 * c_4, c_4 = c_6$. The parameter c_3 is for the linear loss. In order to maintain a tradeoff between linear and quadratic loss, the parameter c_3 is taken as product of the weighting parameters for regularization term, and quadratic loss of the data points. The kernel parameter μ is taken from the set $\{2^{-5}, 2^{-4}, ..., 2^{5}\}$ in all the algorithms. In ATWSVM, LS-ATWSVM, and proposed AULSTSVM the angle parameter c_7 is chosen from $\{0.1, 0.2, ..., 1\}$ [225].

In case of large datasets shown in Table 6.4, the hardware specifications are same as in 3.2.6, the size of training data and universum is 60% and 10% of all samples respectively. The values of $c_i, i = 1, 2$ are fixed as 0.1, $c_i, i = 3, ..., 6$ are fixed as 1, the parameters c_7, ϵ are set as 0.5, and μ is set as 2^4 [158] for all the algorithms. For ADNI datasets, the size of universum is chosen from the set $\{10\%, 15\%\}$. The universum data is generated by random averaging of data points [26, 27], and the same dataset

is used in UTSVM, ULSTSVM, and proposed AULSTSVM.

6.1.6.2 Synthetic dataset

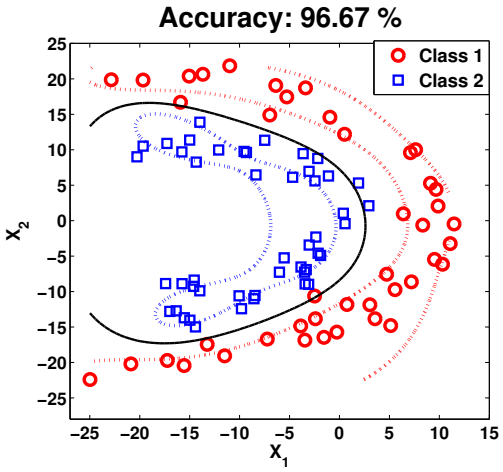
The classifiers constructed by the different algorithms on Half_kernel synthetic dataset [3] are shown in Fig. 6.2. The hyperplanes for the binary classes are shown in red and blue colour, while black coloured hyperplane is the classifier. One can see that instead of twin hyperplanes for the two classes, proposed AULSTSVM constructs one plane for universum (green), and one for classification (black). The AULSTSVM tries to minimize the angle between the hyperplanes, and thus provides prior information to the classifier using the universum plane. In Fig. 6.2, one observation is that the quadratic loss based methods shown in 6.2(a), 6.2(b), and 6.2(c) are showing comparatively better performance than the hinge loss based models in 6.2(d), 6.2(e), and 6.2(f). Among the quadratic loss based methods, the proposed AULSTSVM is showing highest generalization performance shown in 6.2(f). This shows the benefit of introducing the universum plane for providing prior information about the data.

6.1.6.3 Real world benchmark datasets

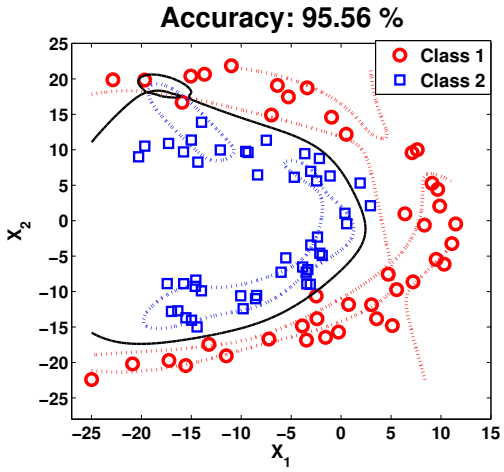
This subsection presents the experimental results with discussion on the performance of the algorithms for both linear and non-linear cases.

I. Linear case:

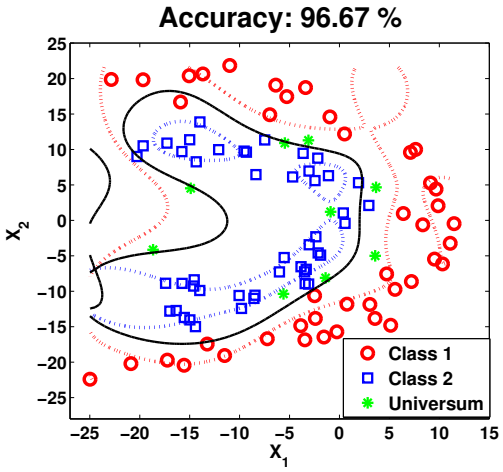
The experimental results for the linear case are shown in Table 6.1. The performance comparison with 6 existing algorithms is based on mean accuracy with standard deviation and training time. One can observe in Table 6.1 that the accuracy of the proposed AULSTSVM is more than existing algorithms in most of the datasets. This is reflected in the average rank of AULSTSVM based on accuracy i.e., 2.2115. It is the least rank among all the methods in Table 6.1. The proposed AULSTSVM is outperforming all the existing algorithms in 10 out of 26 datasets. One can notice that apart from AULSTSVM, other algorithms viz. UTSVM and ULSTSVM performed better than other algorithms in terms of accuracy.



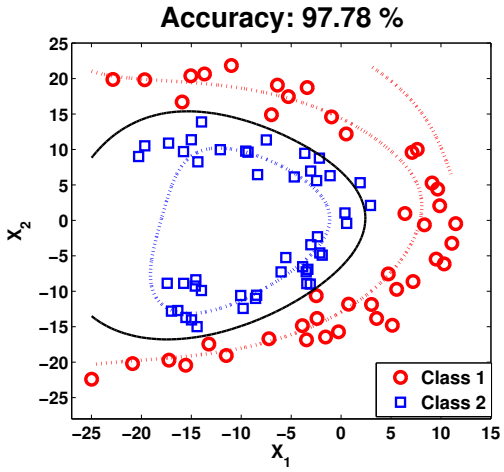
(a) TWSVM



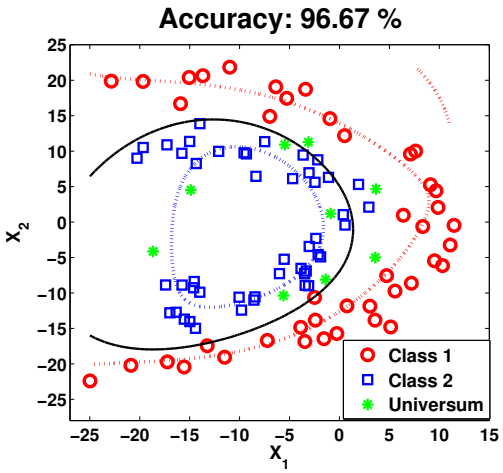
(b) ATWSVM



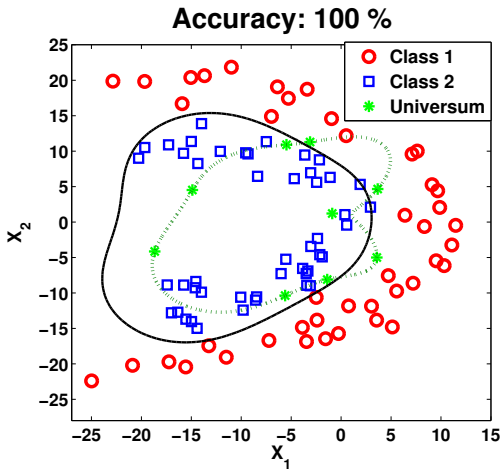
(c) UTSVM



(d) LSTSVM



(e) ULSTSVM



(f) Proposed AULSTSVM

Figure 6.2: Plot showing comparison of classifiers on Half_kernel [3] synthetic dataset using RBF kernel.

Table 6.1: Performance comparison of the proposed AULSTSVM with existing algorithms on real world datasets in linear case.

Dataset (Size)	**TWSVM** Accuracy (%) (c_1) Time (s)	**ATWSVM** Accuracy (%) (c_1, c_4) Time (s)	**UTSVM** Accuracy (%) (c_1, ϵ) Time (s)	**LSTSVM** Accuracy (%) (c_1) Time (ms)	**LS-ATWSVM** Accuracy (%) (c_1, c_4) Time (ms)	**ULSTSVM** Accuracy (%) (c_1, c_3, ϵ) Time (ms)	**AULSTSVM** Accuracy (%) (c_4, c_1, c_7) Time (ms)
Ecoli-0-1_vs_2-3-5 (246 × 7)	90.93±6.79 (10^0) 0.0091	91.03±3.45 (10^{-7}, 0.7) 0.0046	93.47±5.48 (10^1, 0.2) 0.0082	92.53±3.53 (10^1) 0.0656	90.2±3.75 (10^{-5}, 0.3) 0.0626	92.67±3.39 (10^2, 10^5, 0.9) 0.0754	**94.3±6.71** (10^{-5}, 10^{-5}, 0.5) 0.0649
Ecoli-0-1_vs_5 (242 × 6)	92.39±5.56 (10^0) 0.0099	91.77±4.05 (10^{-6}, 0.3) 0.0046	90.87±4.61 (10^4, 0.1) 0.0088	88.22±6.2 (10^1) 0.0759	86.7±6.9 (10^{-9}, 0.1) 0.0541	87.63±5.78 (10^1, 10^{-5}, 0.1) 0.0732	**96.67±3.49** (10^0, 10^2, 0.4) 0.0627
Ecoli-0-1-4-7_vs_5-6 (334 × 6)	95.76±2.71 (10^0) 0.0113	94.05±4.16 (10^{-9}, 0.1) 0.0067	97.01±0.05 (10^0, 0.4) 0.0151	95.76±4.6 (10^{-1}) 0.0623	92.85±5.36 (10^{-9}, 1) 0.0807	95.79±2.68 (10^0, 10^{-5}, 0.9) 0.0806	**97.61±2.47** (10^1, 10^3, 0.8) 0.0592
Ecoli-0-2-3-4_vs_5 (204 × 7)	89±4.18 (10^{-1}) 0.0069	**94.24±6.2** (10^{-6}, 0.5) 0.0047	85.19±7.24 (10^{-1}, 0.1) 0.0102	89±8.94 (10^{-1}) 0.0646	92.24±4.14 (10^{-6}, 0.6) 0.0581	87.33±7.33 (10^{-5}, 10^3, 0.1) 0.0774	94.19±6.28 (10^{-5}, 10^{-5}, 0.1) 0.053
Ecoli-0-2-6-7_vs_3-5 (226 × 7)	93.68±4.09 (10^{-1}) 0.0067	92.17±7.78 (10^{-1}, 0.9) 0.0046	86.88±11.47 (10^{-1}, 0.8) 0.0106	90.12±7.4 (10^0) 0.0587	89.57±11.34 (10^0, 0.9) 0.0676	90.36±6.44 (10^{-1}, 10^{-1}, 0.9) 0.1337	**95.65±4.35** (10^0, 10^3, 0.8) 0.0725
Ecoli-0-3-4-6_vs_5 (206 × 7)	86.05±17.5 (10^0) 0.0063	92.29±9.9 (10^0, 0.5) 0.0041	93.19±2.7 (10^2, 0.3) 0.0108	92.05±4.51 (10^0) 0.0715	90.38±6.74 (10^{-9}, 0.1) 0.0563	89.43±10.85 (10^{-3}, 10^2, 0.1) 0.0752	**97.05±4.45** (10^1, 10^1, 0.4) 0.0575
Ecoli-0-4-6_vs_5 (204 × 6)	**95±3.54** (10^{-1}) 0.0067	91.14±2.3 (10^{-9}, 0.2) 0.004	89.29±7.76 (10^2, 0.6) 0.0086	94±4.18 (10^{-1}) 0.0624	90.1±7.07 (10^{-9}, 0.1) 0.052	94.14±4.09 (10^{-1}, 10^{-5}, 0.1) 0.1042	92.1±7.63 (10^4, 10^2, 0.5) 0.0549
Ecoli-0-6-7_vs_3-5 (224 × 7)	78.18±8.74 (10^1) 0.0069	86.64±6.16 (10^{-6}, 0.7) 0.0045	80.43±16.72 (10^2, 0.1) 0.0081	80.91±19.92 (10^1) 0.0724	87.51±7.2 (10^{-8}, 0.4) 0.0571	92.85±2.48 (10^0, 10^3, 0.4) 0.0776	**96.4±3.79** (10^1, 10^3, 1) 0.0758
Led7digit-0-2-4-5-6-7-8-9_vs_1 (444 × 7)	95±5.43 (10^0) 0.0177	95.05±3.69 (10^0, 0.8) 0.0135	92.8±1.86 (10^1, 0.2) 0.0255	91.82±1.24 (10^0) 0.0853	91.89±1.25 (10^0, 0.9) 0.0579	96.85±2.58 (10^0, 10^{-5}, 0.6) 0.0989	**97.29±2.95** (10^0, 10^1, 0.3) 0.066
Yeast-0-2-5-7-9_vs_3-6-8 (1006 × 8)	95.8±1.49 (10^0) 0.078	96.62±1.5 (10^0, 0.9) 0.0462	**96.82±0.82** (10^0, 0.2) 0.1278	92.61±1.82 (10^0) 0.0838	92.65±2.84 (10^{-1}, 0.9) 0.0675	95.43±0.87 (10^2, 10^2, 0.8) 0.098	96.23±1.9 (10^{-1}, 10^0, 0.9) 0.0724
Yeast-2_vs_4 (516 × 8)	94.92±4.3 (10^2) 0.0165	93.02±1.76 (10^{-9}, 0.7) 0.0171	95.73±1.63 (10^0, 0.5) 0.0231	94.13±3.11 (10^0) 0.0719	91.09±4.42 (10^{-2}, 0.9) 0.063	93.03±2.56 (10^2, 10^2, 0.7) 0.1203	**96.52±2.09** (10^0, 10^0, 0.5) 0.0698
Ecoli-0-1-4-6_vs_5 (282 × 6)	92.75±4.59 (10^1) 0.0106	97.86±3.19 (10^{-1}, 0.8) 0.008	94.38±5.22 (10^0, 0.1) 0.0121	98.57±3.19 (10^{-1}) 0.0716	93.67±4.45 (10^{-9}, 0.1) 0.0749	**99.31±1.54** (10^0, 10^{-5}, 0.6) 0.0775	97.86±3.19 (10^5, 10^4, 0.4) 0.0563
Glass-0-6_vs_5 (110 × 9)	**96.18±5.24** (10^{-5}) 0.0075	94.55±8.13 (10^{-3}, 0.7) 0.0033	87.27±15.21 (10^0, 0.2) 0.0079	88.73±10.32 (10^{-5}) 0.0806	90.91±11.13 (10^{-9}, 0.1) 0.0614	87.27±8.13 (10^{-3}, 10^{-4}, 0.1) 0.1314	89.09±7.61 (10^{-3}, 10^{-3}, 0.6) 0.0795

Abbreviations: s- seconds, ms- milliseconds.

Table 6.1 (contd.)

Dataset (Size)	**TWSVM** Accuracy (%) (c_1) Time (s)	**ATWSVM** Accuracy (%) (c_1, c_4) Time (s)	**UTSVM** Accuracy (%) (c_1, ϵ) Time (s)	**LSTSVM** Accuracy (%) (c_1) Time (ms)	**LS-ATWSVM** Accuracy (%) (c_1, c_4) Time (ms)	**ULSTSVM** Accuracy (%) (c_1, c_3, ϵ) Time (ms)	**AULSTSVM** Accuracy (%) (c_4, c_1, c_7) Time (ms)
Abalone9-18 (732 × 7)	95.07±7.35 (10^2) 0.0441	94.29±6.31 (10^{-9}, 0.6) 0.0258	95.64±4.23 (10^1, 0.5) 0.0556	94.79±6.81 (10^0) 0.0811	94.01±6.02 (10^{-1}, 0.9) 0.0729	95.65±4.81 (10^1, 10^0, 0.7) 0.1108	**95.92±4.69** (10^0, 10^{-1}, 0.7) 0.0954
Pima Indians (770 × 8)	74.16±6.81 (10^0) 0.0201	74.55±5.71 (10^0, 0.6) 0.0121	77.14±2.69 (10^{-1}, 0.7) 0.0311	**79.64±2.81** (10^0) 0.0866	64.42±8.88 (10^0, 0.9) 0.0702	72.99±4.54 (10^0, 10^{-1}, 0.1) 0.103	77.66±2.32 (10^3, 10^3, 0.6) 0.0954
New-thyroid1 (216 × 5)	**98.1±2.61** (10^1) 0.0064	89.78±6.02 (10^0, 0.9) 0.0041	95.37±0.12 (10^0, 0.8) 0.0099	95.32±3.22 (10^{-2}) 0.1015	86.02±6.01 (10^{-1}, 0.9) 0.0638	96.28±2.08 (10^{-2}, 10^2, 0.1) 0.1104	97.23±2.53 (10^2, 10^4, 0.9) 0.0526
Yeast3 (1486 × 8)	**95.27±1.36** (10^0) 0.1635	92.32±4.82 (10^{-1}, 0.4) 0.1273	92.86±3.66 (10^1, 0.7) 0.2889	90.28±6.92 (10^0) 0.082	90.29±6.68 (10^{-1}, 0.9) 0.0746	92.86±2.89 (10^0, 10^{-5}, 0.7) 0.1504	92.32±3.27 (10^2, 10^2, 0.4) 0.0753
Yeast1vs7 (460 × 8)	93±3.55 (10^0) 0.0131	**93.48±3.44** (10^{-2}, 0.2) 0.0085	**93.48±3.44** (10^0, 0.4) 0.0215	93.45±3.41 (10^0) 0.0688	92.17±3.64 (10^{-2}, 0.9) 0.068	54.35±24.35 (10^2, 10^2, 0.8) 0.084	**93.48±3.44** (10^{-1}, 10^{-2}, 1) 0.0653
Ecoli0137vs26 (312 × 7)	94.19±4.21 (10^{-1}) 0.0105	94.21±8.04 (10^{-1}, 0.8) 0.0076	**96.79±3.23** (10^{-1}, 0.1) 0.0122	96.77±2.28 (10^{-1}) 0.0658	88.45±8.73 (10^{-1}, 0.9) 0.077	95.5±6.69 (10^{-1}, 10^{-5}, 0.6) 0.1101	95.5±4.9 (10^{-1}, 10^{-1}, 0.5) 0.0633
Glass-0-1-6_vs_2 (194 × 9)	88.42±6.86 (10^0) 0.0088	91.79±5.91 (10^{-7}, 0.5) 0.0045	**93.89±4.15** (10^0, 0.1) 0.0089	93.68±4.4 (10^0) 0.0742	**93.89±4.15** (10^{-9}, 1) 0.0789	**93.89±4.15** (10^0, 10^5, 0.9) 0.0856	**93.89±4.15** (10^2, 10^2, 0.3) 0.07
Monk2 (602 × 7)	**68.21±10.15** (10^0) 0.0105	45.24±11.44 (10^{-6}, 0.8) 0.0092	58.79±13.85 (10^1, 0.4) 0.0225	56.77±16.17 (10^0) 0.0882	64.13±6.25 (10^{-9}, 0.1) 0.085	54.13±15.02 (10^1, 10^{-5}, 0.9) 0.0853	64.13±6.25 (10^{-5}, 10^{-2}, 0.1) 0.0632
Australian credit (692 × 14)	**86.34±4.65** (10^0) 0.018	83.52±2.25 (10^0, 0.8) 0.0106	84.68±6.54 (10^0, 0.1) 0.0298	85.78±7.32 (10^0) 0.0901	71.66±2.94 (10^{-9}, 0.9) 0.0853	85.83±6.28 (10^1, 10^3, 0.8) 0.1145	85.25±3.77 (10^{-3}, 10^0, 0.7) 0.084
Ripley (1252 × 2)	88.15±2.59 (10^1) 0.0391	87.22±2.49 (10^0, 0.9) 0.0154	87.06±1.34 (10^1, 0.4) 0.0747	**89.1±1.97** (10^{-5}) 0.0992	68.69±4.03 (10^{-9}, 0.9) 0.0656	88.5±2.38 (10^1, 10^0, 0.4) 0.1015	87.54±3.33 (10^{-5}, 10^{-4}, 0.1) 0.0823
Transfusion (750 × 4)	70.3±15.98 (10^0) 0.0217	82.67±14.02 (10^{-6}, 0.9) 0.0215	82.13±14.38 (10^0, 0.1) 0.0356	82.01±14.55 (10^1) 0.0856	82.4±14.19 (10^{-6}, 0.6) 0.0569	79.47±23.24 (10^0, 10^{-1}, 0.9) 0.0937	**84.53±14.78** (10^2, 10^5, 0.3) 0.0712
Votes (436 × 16)	**97.22±1.05** (10^0) 0.0088	90.41±7.91 (10^{-9}, 0.6) 0.0079	93.59±4.35 (10^0, 0.9) 0.0123	94.89±5.05 (10^{-5}) 0.1304	60.06±3.83 (10^{-9}, 1) 0.1112	84.38±6.28 (10^1, 10^{-5}, 0.2) 0.1921	94.97±4.92 (10^0, 10^1, 0.2) 0.0821
Vowel (990 × 10)	88.86±5.46 (10^{-1}) 0.0622	89.49±6.09 (10^0, 0.4) 0.0523	92.12±6.9 (10^2, 0.4) 0.112	91.08±2.7 (10^0) 0.1129	90.1±1.81 (10^{-1}, 0.8) 0.1125	**92.12±6.68** (10^0, 10^{-1}, 0.5) 0.1237	91.72±7.09 (10^4, 10^4, 0.5) 0.0755
Average accuracy	89.73	89.21	89.5	89.69	85.62	88	**92.12**
Average rank	3.8846	4.2885	3.8654	4.2692	5.6154	3.8654	**2.2115**

In Table 6.1, the average ranks based on accuracy of UTSVM and ULSTSVM are the same i.e., 3.8654. This can be attributed to the use of universum data in both of these algorithms. In terms of computational efficiency, AULSTSVM takes very less training time in comparison to TWSVM, ATWSVM, UTSVM, and ULSTSVM in Table 6.1. This is because TWSVM, ATWSVM and UTSVM solve computationally intensive QPPs, while AULSTSVM solves system of linear equations. The training time of the proposed AULSTSVM is lesser than ULSTSVM due to the introduction of linear loss in AULSTSVM. The difference in training time is clearly visible in the large sample datasets viz. Yeast-0-2-5-7-9_vs_3-6-8, Yeast3 and Ripley. In these datasets, the training time of proposed AULSTSVM is lesser than LSTSVM algorithm also. It shows the benefit of the proposed algorithm which includes universum data for prior information, but with lesser computation time.

II. Non-linear case:

In non-linear case also, the proposed AULSTSVM outperforms the existing algorithms in terms of accuracy in Table 6.2. The proposed AULSTSVM performed better than existing algorithms in 11 out of 26 datasets. This shows the superiority of proposed AULSTSVM over the other algorithms. Among the 7 algorithms, proposed AULSTSVM achieves lowest rank on accuracy i.e., 2.2692. The second algorithm with better performance in terms of accuracy is UTSVM with a rank of 3.4615. This shows the superiority of universum based SVM algorithms. One can observe that in the non-linear case also, the training time of AULSTSVM is very less as compared to TWSVM, ATWSVM, and UTSVM. However, the training time of AULSTSVM is comparable to LSTSVM. This is due to the overhead of calculating the universum kernel matrix in proposed AULSTSVM. In comparison to ULSTSVM, the training time of AULSTSVM is very less. This means that proposed AULSTSVM utilizes the universum data using kernel matrix like in ULSTSVM, but with reduced computation cost. Thus, non-linear AULSTSVM also removes the drawback of increased computation time due to addition of universum data [20]. The computation time of proposed AULSTSVM is comparable to the existing algorithms in the non-linear case, because most computations are involved in the construction of kernel matrices.

Table 6.2: Performance comparison of the proposed AULSTSVM with existing algorithms on real world datasets for non-linear case.

Dataset (Size)	**TWSVM** Accuracy (%) (c_1, μ) Time (s)	**ATWSVM** Accuracy (%) (c_1, c_4, μ) Time (s)	**UTSVM** Accuracy (%) (c_1, μ, ϵ) Time (s)	**LSTSVM** Accuracy (%) (c_1, μ) Time (ms)	**LS-ATWSVM** Accuracy (%) (c_1, c_4, μ) Time (s)	**ULSTSVM** Accuracy (%) $(c_1 = c_3, \epsilon, \mu)$ Time (s)	**AULSTSVM** Accuracy (%) $(c_4 = c_1, c_7, \mu)$ Time (s)
Ecoli-0-1_vs_2-3-5 (246 × 7)	92.63±7.25 $(10^{-5}, 2^5)$ 0.0204	91.83±6.41 $(10^0, 0.1, 2^5)$ 0.0183	93.43±6.21 $(10^{-5}, 2^5, 0.1)$ 0.0227	91.03±5.99 $(10^{-5}, 2^4)$ 0.0117	90.2±3.75 $(10^0, 0.1, 2^5)$ 0.0132	93.47±5.48 $(10^3, 0.3, 2^5)$ 0.0166	**94.27±4.72** $(10^3, 0.3, 2^5)$ 0.0123
Ecoli-0-1_vs_5 (242 × 6)	95.87±4.17 $(10^{-5}, 2^5)$ 0.0211	94.2±2.31 $(10^{-3}, 0.6, 2^5)$ 0.0196	90.93±3.37 $(10^{-2}, 2^4, 0.4)$ 0.024	91.8±6.38 $(10^3, 2^5)$ 0.0144	92.57±4.55 $(10^{-1}, 0.8, 2^5)$ 0.0135	**96.7±3.48** $(10^{-1}, 0.6, 2^5)$ 0.0164	**96.7±3.48** $(10^0, 0.1, 2^5)$ 0.012
Ecoli-0-1-4-7_vs_5-6 (334 × 6)	97.61±2.47 $(10^{-3}, 2^5)$ 0.0324	97.01±2.99 $(10^0, 0.6, 2^5)$ 0.0299	96.4±3.29 $(10^{-1}, 2^5, 0.7)$ 0.0374	95.81±4.53 $(10^1, 2^5)$ 0.0216	93.44±5.68 $(10^0, 0.9, 2^5)$ 0.022	97.01±2.99 $(10^0, 0.6, 2^5)$ 0.0268	**98.22±2.64** $(10^2, 0.1, 2^5)$ 0.0217
Ecoli-0-2-3-4_vs_5 (204 × 7)	96.05±2.21 $(10^{-5}, 2^4)$ 0.0178	**99.05±2.13** $(10^{-1}, 0.1, 2^5)$ 0.0128	99±2.24 $(10^{-3}, 2^4, 0.4)$ 0.0189	96.05±2.21 $(10^{-5}, 2^4)$ 0.0115	85.52±11.02 $(10^{-1}, 0.6, 2^5)$ 0.0084	98±2.74 $(10^0, 0.3, 2^5)$ 0.013	98.1±4.26 $(10^{-2}, 0.4, 2^5)$ 0.0087
Ecoli-0-2-6-7_vs_3-5 (226 × 7)	95.65±4.35 $(10^{-3}, 2^5)$ 0.0229	95.65±4.35 $(10^0, 0.7, 2^5)$ 0.0182	92.96±4.99 $(10^{-3}, 2^4, 0.5)$ 0.0277	91.15±6.15 $(10^{-2}, 2^5)$ 0.0113	93.04±6.59 $(10^{-1}, 0.8, 2^5)$ 0.0098	91.26±8.11 $(10^4, 0.2, 2^5)$ 0.0148	**96.52±3.64** $(10^1, 0.1, 2^5)$ 0.0101
Ecoli-0-3-4-6_vs_5 (206 × 7)	95.14±3.54 $(10^{-5}, 2^5)$ 0.0135	**98.05±2.67** $(10^{-1}, 0.8, 2^5)$ 0.0158	95.1±6.13 $(10^{-1}, 2^5, 0.7)$ 0.0197	96.14±4 $(10^{-5}, 2^4)$ 0.0117	91.33±7.08 $(10^{-1}, 0.1, 2^4)$ 0.0113	96.1±4.16 $(10^{-3}, 0.8, 2^5)$ 0.0132	97.05±2.7 $(10^0, 0.2, 2^5)$ 0.0089
Ecoli-0-4-6_vs_5 (204 × 6)	90.14±5.06 $(10^{-5}, 2^4)$ 0.0192	95±5 $(10^{-1}, 0.9, 2^5)$ 0.0169	95±7.07 $(10^{-3}, 2^4, 0.4)$ 0.0198	91.14±6.54 $(10^{-5}, 2^4)$ 0.0108	90.14±3.63 $(10^{-1}, 0.9, 2^5)$ 0.0103	94.1±4.21 $(10^1, 0.9, 2^5)$ 0.0154	**96±5.48** $(10^{-5}, 0.1, 2^5)$ 0.0085
Ecoli-0-6-7_vs_3-5 (224 × 7)	92.06±5.66 $(10^{-3}, 2^4)$ 0.0193	**94.7±3.62** $(10^0, 0.9, 2^5)$ 0.0175	94.66±3.72 $(10^{-2}, 2^5, 0.2)$ 0.0219	91.07±6.36 $(10^{-5}, 2^5)$ 0.0109	90.24±4.68 $(10^{-1}, 0.9, 2^5)$ 0.0093	93.75±4.03 $(10^{-2}, 0.1, 2^5)$ 0.0145	94.66±3.72 $(10^1, 0.2, 2^5)$ 0.01
Led7digit-0-2-4-5-6-7-8-9_vs_1 (444 × 7)	94.64±5.1 $(10^{-4}, 2^1)$ 0.0529	96.39±3.45 $(10^{-9}, 0.8, 2^4)$ 0.0476	96.86±2.54 $(10^{-2}, 2^1, 0.9)$ 0.0596	96.39±3.41 $(10^{-1}, 2^3)$ 0.0363	91.89±1.25 $(10^{-9}, 0.5, 2^{-5})$ 0.036	95.95±2.94 $(10^{-1}, 0.1, 2^2)$ 0.0442	**97.75±2.25** $(10^3, 0.2, 2^1)$ 0.0376
Yeast-0-2-5-7-9_vs_3-6-8 (1006 × 8)	**97.42±1.34** $(10^{-1}, 2^{-2})$ 0.2386	95.43±1.5 $(10^{-5}, 0.9, 2^{-3})$ 0.2398	96.23±0.81 $(10^{-1}, 2^1, 0.3)$ 0.2991	93.84±1.62 $(10^{-1}, 2^2)$ 0.166	92.85±1.89 $(10^{-5}, 0.9, 2^4)$ 0.1588	94.63±1.32 $(10^0, 0.7, 2^{-2})$ 0.2229	97.22±1.29 $(10^5, 0.4, 2^{-1})$ 0.1914
Yeast-2_vs_4 (516 × 8)	95.36±2.9 $(10^{-2}, 2^0)$ 0.0613	92.26±3.83 $(10^{-1}, 0.9, 2^{-1})$ 0.0594	96.12±1.36 $(10^{-1}, 2^0, 0.5)$ 0.076	95.36±3.21 $(10^0, 2^0)$ 0.0568	88.79±4.12 $(10^{-5}, 0.9, 2^0)$ 0.0441	94.97±2.9 $(10^{-2}, 0.1, 2^0)$ 0.0588	**97.29±1.71** $(10^0, 0.7, 2^0)$ 0.0511
Ecoli-0-1-4-6_vs_5 (282 × 6)	98.62±3.08 $(10^{-3}, 2^5)$ 0.0224	**99.31±1.54** $(10^{-1}, 0.7, 2^5)$ 0.0236	98.62±3.08 $(10^{-4}, 2^5, 0.7)$ 0.0282	**99.31±1.54** $(10^{-3}, 2^5)$ 0.0128	85.1±13.23 $(10^{-1}, 0.1, 2^5)$ 0.0134	98.62±3.08 $(10^{-2}, 0.1, 2^5)$ 0.0161	99.31±1.54 $(10^4, 0.6, 2^5)$ 0.0155
Glass-0-6_vs_5 (110 × 9)	92.73±7.61 $(10^{-4}, 2^0)$ 0.0103	92.73±7.61 $(10^{-7}, 0.9, 2^1)$ 0.0067	**94.55±8.13** $(10^1, 2^4, 0.1)$ 0.0113	92.73±7.61 $(10^{-5}, 2^0)$ 0.0036	78.18±34.38 $(10^{-9}, 0.2, 2^{-5})$ 0.004	90.91±9.09 $(10^{-5}, 0.4, 2^0)$ 0.0048	90.91±9.09 $(10^{-5}, 0.2, 2^{-1})$ 0.0028

Abbreviations: s- seconds, ms- milliseconds.

Table 6.2 (contd.)

Dataset (Size)	**TWSVM** Accuracy (%) (c_1, μ) Time (s)	**ATWSVM** Accuracy (%) (c_1, c_4, μ) Time (s)	**UTSVM** Accuracy (%) (c_1, μ, ϵ) Time (s)	**LSTSVM** Accuracy (%) (c_1, μ) Time (ms)	**LS-ATWSVM** Accuracy (%) (c_1, c_4, μ) Time (s)	**ULSTSVM** Accuracy (%) $(c_1 = c_3, \epsilon, \mu)$ Time (s)	**AULSTSVM** Accuracy (%) $(c_4 = c_1, c_7, \mu)$ Time (s)
Abalone9-18 (732 × 7)	93.18±4.61 $(10^{-2}, 2^{0})$ 0.1214	93.75±7.32 $(10^{-9}, 0.8, 2^{4})$ 0.1216	94.01±4.65 $(10^{-1}, 2^{0}, 0.9)$ 0.1527	92.9±2.96 $(10^{-1}, 2^{1})$ 0.0883	94.01±6.54 $(10^{-9}, 0.7, 2^{5})$ 0.0839	94.29±6.74 $(10^{-1}, 0.4, 2^{0})$ 0.1103	**95.1±4.64** $(10^{2}, 0.4, 2^{-1})$ 0.1019
Pima Indians (770 × 8)	75.06±5.15 $(10^{-4}, 2^{5})$ 0.1268	71.69±5.46 $(10^{-1}, 0.2, 2^{5})$ 0.1171	75.06±2.82 $(10^{-3}, 2^{5}, 0.9)$ 0.1538	68.83±6.3 $(10^{3}, 2^{3})$ 0.1062	68.05±9.74 $(10^{-2}, 0.5, 2^{5})$ 0.0941	71.95±6.53 $(10^{-4}0.6, 2^{5})$ 0.1252	**75.58±2.32** $(10^{-5}, 1, 2^{5})$ 0.1121
New-thyroid1 (216 × 5)	97.19±2.57 $(10^{-2}, 2^{4})$ 0.0163	97.19±2.57 $(10^{-1}, 0.9, 2^{3})$ 0.0125	**99.05±2.13** $(10^{-3}, 2^{5}, 0.3)$ 0.0179	96.32±3.82 $(10^{0}, 2^{4})$ 0.01	90.69±4.71 $(10^{0}, 0.9, 2^{4})$ 0.0092	94.33±7.83 $(10^{3}, 0.8, 2^{3})$ 0.0175	94.42±3.9 $(10^{0}, 0.3, 2^{3})$ 0.0095
Yeast3 (1486 × 8)	94.61±2.49 $(10^{-2}, 2^{0})$ 0.5232	90.57±5.83 $(10^{-3}, 0.6, 2^{2})$ 0.5205	94.61±2.66 $(10^{3}, 2^{2}, 0.4)$ 0.7458	94.34±2.07 $(10^{-1}, 2^{1})$ 0.3651	89.62±6.43 $(10^{0}, 0.9, 2^{-1})$ 0.3628	94.07±2.89 $(10^{2}, 0.9, 2^{-1})$ 0.4955	**95.15±1.82** $(10^{4}, 0.2, 2^{-1})$ 0.4305
Yeast1vs7 (460 × 8)	92.61±3.3 $(10^{-1}, 2^{-2})$ 0.0514	92.61±3.95 $(10^{-3}, 0.9, 2^{-2})$ 0.0487	91.3±1.54 $(10^{-1}, 2^{-2}, 0.1)$ 0.0627	92.61±3.95 $(10^{0}, 2^{-1})$ 0.0371	91.74±3.22 $(10^{-8}, 0.9, 2^{5})$ 0.0363	90±2.92 $(10^{-2}, 0.3, 2^{-2})$ 0.0467	**93.48±3.44** $(10^{1}, 0.2, 2^{-1})$ 0.04
Ecoli0137vs26 (312 × 7)	95.52±3.65 $(10^{-2}, 2^{-2})$ 0.0273	91.09±9.63 $(10^{-8}, 0.9, 2^{2})$ 0.0263	75.6±8.81 $(10^{2}, 2^{3}, 0.3)$ 0.0329	93.61±5.05 $(10^{0}, 2^{-1})$ 0.0156	88.43±10.38 $(10^{-5}, 0.9, 2^{4})$ 0.0156	94.23±2.69 $(10^{-2}, 0.1, 2^{2})$ 0.0196	**97.48±4.08** $(10^{2}, 0.3, 2^{-2})$ 0.0183
Glass-0-1-6_vs_2 (194 × 9)	91.84±4.34 $(10^{-4}, 2^{1})$ 0.0172	92.89±5.66 $(10^{-6}, 0.2, 2^{0})$ 0.0158	91.84±4.34 $(10^{-4}, 2^{1}, 0.1)$ 0.0193	**93.89±4.15** $(10^{-5}, 2^{-5})$ 0.01	57.89±48.45 $(10^{-9}, 0.1, 2^{-5})$ 0.011	91.84±5.72 $(10^{-2}, 0.9, 2^{-1})$ 0.0141	**93.89±4.15** $(10^{-5}, 0.6, 2^{-3})$ 0.0078
Monk2 (602 × 7)	**64.13±6.25** $(10^{-5}, 2^{-5})$ 0.0743	**64.13±6.25** $(10^{-9}, 0.1, 2^{-5})$ 0.0751	**64.13±6.25** $(10^{-5}, 2^{-5}, 0.1)$ 0.0918	**64.13±6.25** $(10^{-5}, 2^{-5})$ 0.0614	58.88±14.46 $(10^{-2}, 0.9, 2^{2})$ 0.0651	**64.13±6.25** $(10^{-5}, 0.1, 2^{-5})$ 0.0738	**64.13±6.25** $(10^{-5}, 0.1, 2^{-5})$ 0.0676
Australian credit (692 × 14)	83.52±4.46 $(10^{-2}, 2^{3})$ 0.0989	85.25±6.45 $(10^{-2}, 0.9, 2^{5})$ 0.0938	83.52±4.46 $(10^{-2}, 2^{3}, 0.1)$ 0.1251	**85.83±6.46** $(10^{0}, 2^{5})$ 0.0848	71.69±2.88 $(10^{0}, 0.9, 2^{3})$ 0.0799	82.66±2.25 $(10^{-1}, 0.1, 2^{5})$ 0.1036	84.96±6.48 $(10^{-5}, 0.1, 2^{2})$ 0.0959
Ripley (1252 × 2)	89.46±1.88 $(10^{0}, 2^{-1})$ 0.3209	89.46±1.05 $(10^{-7}, 0.9, 2^{-3})$ 0.2828	90.89±1.22 $(10^{-1}, 2^{-1}, 0.9)$ 0.4103	**91.22±1.76** $(10^{0}, 2^{-1})$ 0.2551	60.54±9.74 $(10^{0}, 0.9, 2^{-2})$ 0.258	90.42±2.19 $(10^{-2}, 0.9, 2^{0})$ 0.3299	90.74±0.71 $(10^{-3}, 0.4, 2^{-2})$ 0.2947
Transfusion (750 × 4)	81.6±13.84 $(10^{-4}, 2^{-5})$ 0.1216	82.4±13.87 $(10^{-3}, 0.6, 2^{5})$ 0.1196	79.2±13.49 $(10^{-1}, 2^{0}, 0.4)$ 0.1566	82.4±14.19 $(10^{3}, 2^{3})$ 0.0905	75.47±10.48 $(10^{-3}, 0.6, 2^{4})$ 0.0917	**84.53±13.08** $(10^{-1}, 0.6, 2^{4})$ 0.1139	81.6±13.84 $(10^{-5}, 0.1, 2^{-5})$ 0.109
Votes (436 × 16)	94.5±4.7 $(10^{-1}, 2^{5})$ 0.0472	93.59±2.45 $(10^{-3}, 0.7, 2^{5})$ 0.0399	**94.97±3.72** $(10^{-1}, 2^{3}, 0.6)$ 0.0558	94.97±4.92 $(10^{0}, 2^{4})$ 0.0357	82.56±6.73 $(10^{0}, 0.9, 2^{4})$ 0.0355	94.5±4.7 $(10^{-3}, 0.1, 2^{4})$ 0.0416	94.05±4.41 $(10^{-3}, 0.5, 2^{2})$ 0.038
Vowel (990 × 10)	93.74±8.88 $(10^{-1}, 2^{5})$ 0.2341	94.55±4.26 $(10^{-4}, 0.8, 2^{5})$ 0.2234	**96.97±3.98** $(10^{-1}, 2^{3}, 0.5)$ 0.3008	93.74±10.82 $(10^{-1}, 2^{4})$ 0.1579	87.88±8.51 $(10^{-4}, 0.4, 2^{4})$ 0.1627	94.95±5.1 $(10^{1}, 0.6, 2^{2})$ 0.2127	93.74±6.04 $(10^{2}, 0.1, 2^{2})$ 0.1894
Average accuracy	91.57	91.57	91.19	91.02	83.87	91.44	**92.63**
Average rank	3.9615	3.4808	3.4615	4.1538	6.5385	4.1346	**2.2692**

6.1.6.4 Statistical analysis

To verify the statistical significance of the results, we perform statistical tests viz. Friedman test [172], and Nemenyi posthoc test in the linear and non-linear cases.

I. Linear case:

Using Table 6.1, we perform the Friedman test and the corresponding post hoc test on 7 algorithms and 26 datasets. First, we assume that there is no difference between the methods as the null hypothesis. χ_F^2 for Friedman test is calculated using average ranks from Table 6.1 to be 33.5038.

The F_F value is calculated as

$$F_F = \frac{(26-1)(33.5038)}{26 \times (7-1) - 33.5038} = 6.8377.$$

Here, the F-distribution has $\big(7-1, (7-1)(26-1)\big) = (6, 150)$ degrees of freedom. Thus, for the significance level at $\alpha = 0.05$, the critical value for $F(6, 150)$ is 2.1595. Since $F_F = 6.8377 > 2.1595$, we reject the null hypothesis.

Now, to check the pairwise difference between the proposed AULSTSVM and existing algorithms, we use the Nemenyi posthoc test. For significant pairwise difference between the methods at significance level of $\alpha = 0.10$, the average ranks of the methods shown in Table 6.1 should differ by atleast $2.693\sqrt{\frac{7(7+1)}{6\times 26}} = 1.6135$. The pairwise difference between the methods is shown in Table 6.3. It can be stated that in the linear case, the proposed AULSTSVM is significantly better than TWSVM, ATWSVM, UTSVM, LSTSVM, LS-ATWSVM, and ULSTSVM algorithms.

Table 6.3: Pairwise significance of the proposed AULSTSVM with existing algorithms.

Linear	**TWSVM**	**ATWSVM**	**UTSVM**	**LSTSVM**	**LS-ATWSVM**	**ULSTSVM**
Proposed AULSTSVM	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Non-linear	**TWSVM**	**ATWSVM**	**UTSVM**	**LSTSVM**	**LS-ATWSVM**	**ULSTSVM**
Proposed AULSTSVM	**Yes**	No	No	**Yes**	**Yes**	**Yes**

II. Non-linear case:

First, we calculate the value of χ_F^2 for Friedman test using Table 6.2 as 55.9464.

The F_F value is calculated as

$$F_F = \frac{(26-1)(55.9464)}{26 \times (7-1) - 55.9464} = 13.9791.$$

Since $F_F = 13.9791 > 2.1595$, we reject the null hypothesis. Again, for checking pairwise difference between proposed AULSTSVM and existing algorithms, we perform the Nemenyi post hoc test. The pairwise difference between the methods for the non-linear case is shown in Table 6.3. It can be stated that the proposed AULSTSVM is significantly better than TWSVM, LSTSVM, LS-ATWSVM, and ULSTSVM.

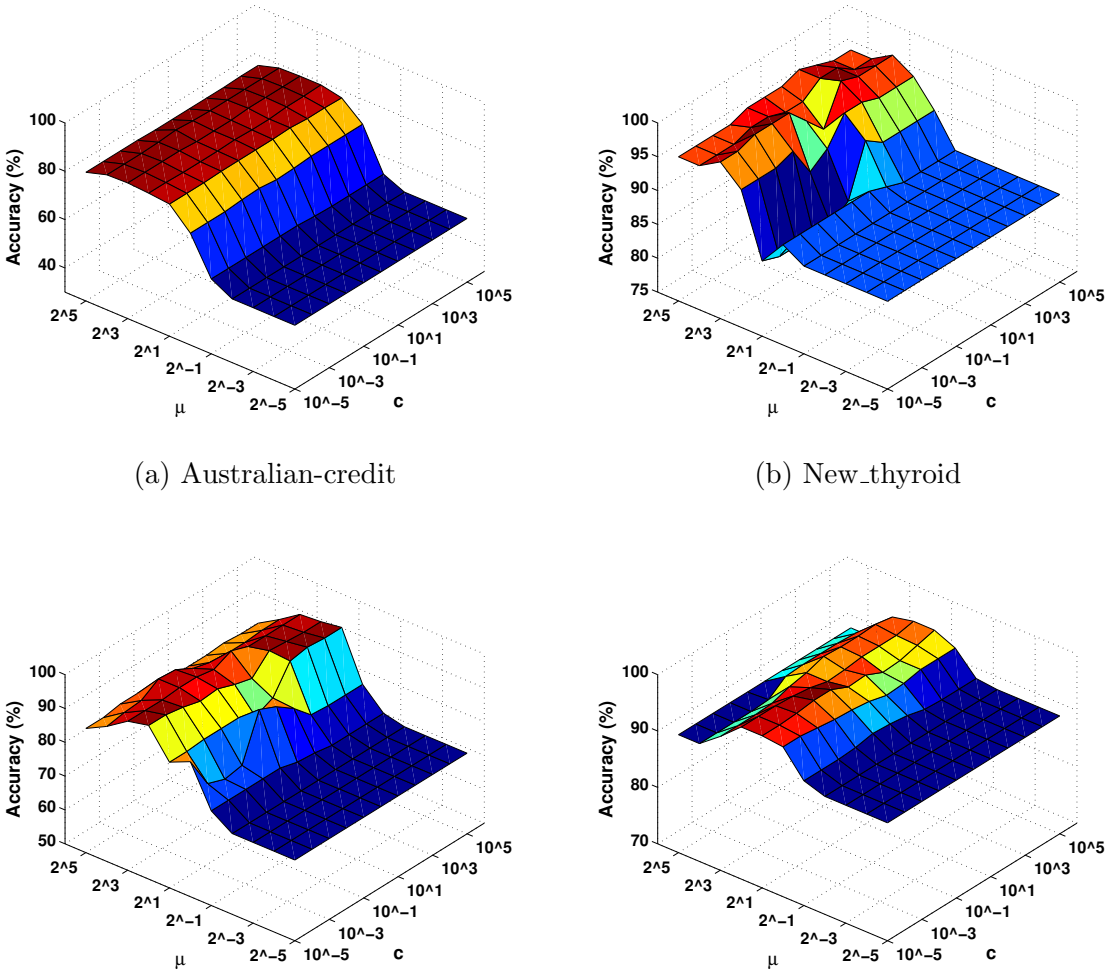
6.1.6.5 Insensitivity analysis

The proposed AULSTSVM involves many hyperparameters viz. penalty parameters $c_i, i = 1, ..., 6$, angle parameter c_7, and kernel parameter μ. The performance of proposed non-linear AULSTSVM for varying values of the user defined parameters c i.e., penalty parameter, and μ is shown in Fig. 6.3.

The plots for classification accuracy on various parameters and datasets are shown in Fig. 6.3. It is observable that the accuracy of proposed AULSTSVM increases for higher values of μ. However, the value of c does not have any significant effect on the accuracy of the model. This is due to the fact that in non-linear case, the value of μ in the RBF kernel significantly affects the complexity of the classifier.

6.1.7 Large scale datasets

To further analyze the computation cost of the proposed approach, experiments are performed on large scale NDC datasets [186]. A rectangular kernel [26] matrix is used in all the algorithms with 10% of the samples of the binary classes. The classification accuracy, and training time of different algorithms are shown in Table 6.4. For the comparison on large datasets, LSTSVM and ULSTSVM are chosen because of their better performance in Table 6.2 w.r.t. accuracy and computation time. It is observable in Table 6.4 that our proposed AULSTSVM is performing better than the other algorithms on most datasets.



(a) Australian-credit

(b) New_thyroid

(c) Votes

(d) Vowel

Figure 6.3: Insensitivity performance of proposed AULSTSVM on the user defined parameters c and μ.

Table 6.4: Performance comparison of the proposed AULSTSVM on large scale datasets.

Dataset	**LSTSVM** Accuracy (%) Time (s)	**ULSTSVM** Accuracy (%) Time (s)	**AULSTSVM** Accuracy (%) Time (s)
NDC-2.5k	61.3 0.3167	59.7 0.33	**61.5** 0.2983
NDC-5k	59.3 1.271	56.9 1.7791	**60.6** 1.1822
NDC-10k	66.9 6.1947	59.28 8.5494	**69.93** 5.5014
NDC-15k	69.42 16.9719	45.7 24.2488	**72.98** 13.9909
NDC-20k	71.65 33.1673	48.08 51.1297	**75.59** 27.8415
NDC-25k	75.68 58.081	62.14 90.2783	**80.65** 48.2124
NDC-30k	78.03 109.999	63.39 161.552	**82.97** 101.05
NDC-35k	78.87 130.889	42.74 241.996	**83.26** 111.261
NDC-40k	83.78 195.487	81.5 393.959	**87.23** 179.878
NDC-45k	83.07 266.736	65.36 409.817	**87.79** 209.356
NDC-50k	85.34 356.511	57.99 538.553	**87.65** 281.122
NDC-55k	85.46 445.58	81.62 908.097	**89.71** 387.529
NDC-60k	85.35 555.833	69.98 931.383	**92.86** 435.448
NDC-70k	88.68 1007.36	69.18 1419.21	**96.93** 661.052
NDC-80k	88.09 1992.52	71.28 2195.76	**93.36** 995.134

Abbreviations: s- seconds.

In terms of computation time, the proposed AULSTSVM is better than LSTSVM and ULSTSVM in all the datasets. This is illustrated in Fig. 6.4, showing the comparative advantage of the proposed approach. One can observe that for very large datasets, such as NDC-60k, NDC-70k, and NDC-80k, the training time of proposed AULSTSVM is very less as compared to LSTSVM and ULSTSVM. This is due to the matrix inverse calculation for universum data in the proposed AULSTSVM. The complexity of constructing kernel matrix of m samples is $O(m^2)$, while inverse calculation is $O(m^3)$. Consequently, for large datasets the computation time for inverse calculation is very high. However, the size of universum matrix in the proposed AULSTSVM is very less size in comparison to the matrices of data points in LSTSVM. Therefore, the proposed AULSTSVM algorithm is suitable for large scale datasets.

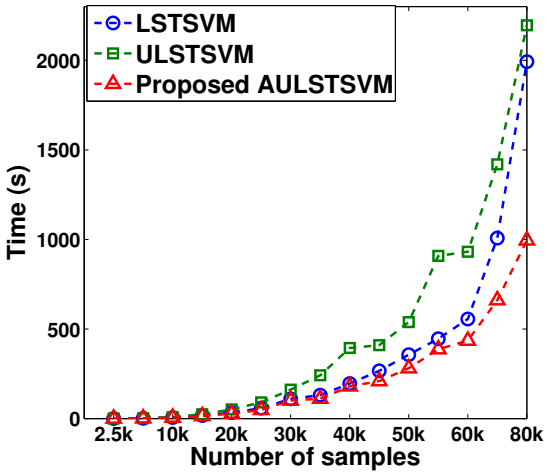
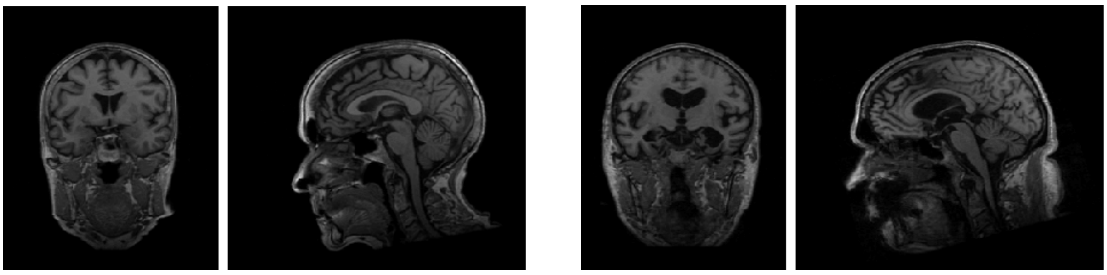


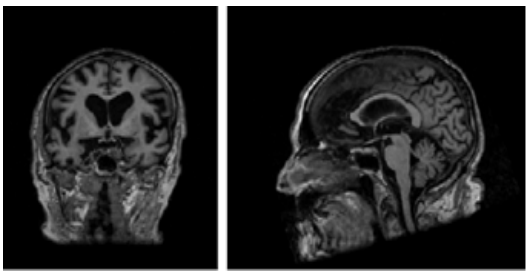
Figure 6.4: Plot showing comparison of LSTSVM, ULSTSVM, and proposed AULSTSVM algorithm on training time for large scale datasets.

6.1.8 Application to Alzheimer's disease

In order to verify the applicability of proposed AULSTSVM on real world applications, we present an application on classification of Alzheimer's disease data. We classify three classes namely control normal (CN), mild cognitive impairment (MCI), and Alzheimer's disease (AD) shown in Fig. 6.5. A total of 150 T1-weighted structural MRI images are used from ADNI database with the same specifications as in



(a) CN (b) MCI



(c) AD

Figure 6.5: Structural MRI images from ADNI database showing coronol (left image) and saggital (right image) view of head in CN, MCI, and AD subjects.

subsection 4.2.2. We used three types of VolBM features viz. SCV, WM, and CT as specified in subsection 4.2.4.

We compared the performance of the proposed AULSTSVM with TWSVM, UTSVM, LSTSVM, and ULSTSVM algorithms using linear kernel [2]. We performed three types of classifications viz. CN vs AD, CN vs MCI, and MCI vs AD as shown in Fig. 6.6. In case of CN vs AD, one can see in Fig. 6.6(a) that the proposed AULSTSVM achieves highest 95% accuracy for SCV features. For CN vs MCI, the proposed AULSTSVM obtains 84.21% accuracy, which is more than the best accuracy of other algorithms. However, in case of MCI vs AD, highest accuracy of 68.42% is obtained by ULSTSVM, as compared to 63.16% accuracy of AULSTSVM. This is because MCI vs AD is a relatively difficult task for classification resulting in lower accuracy [2, 98].

In the next section, we discuss a novel and efficient formulation for least squares based SVM algorithm with universum data, using parametric-margin based approach.

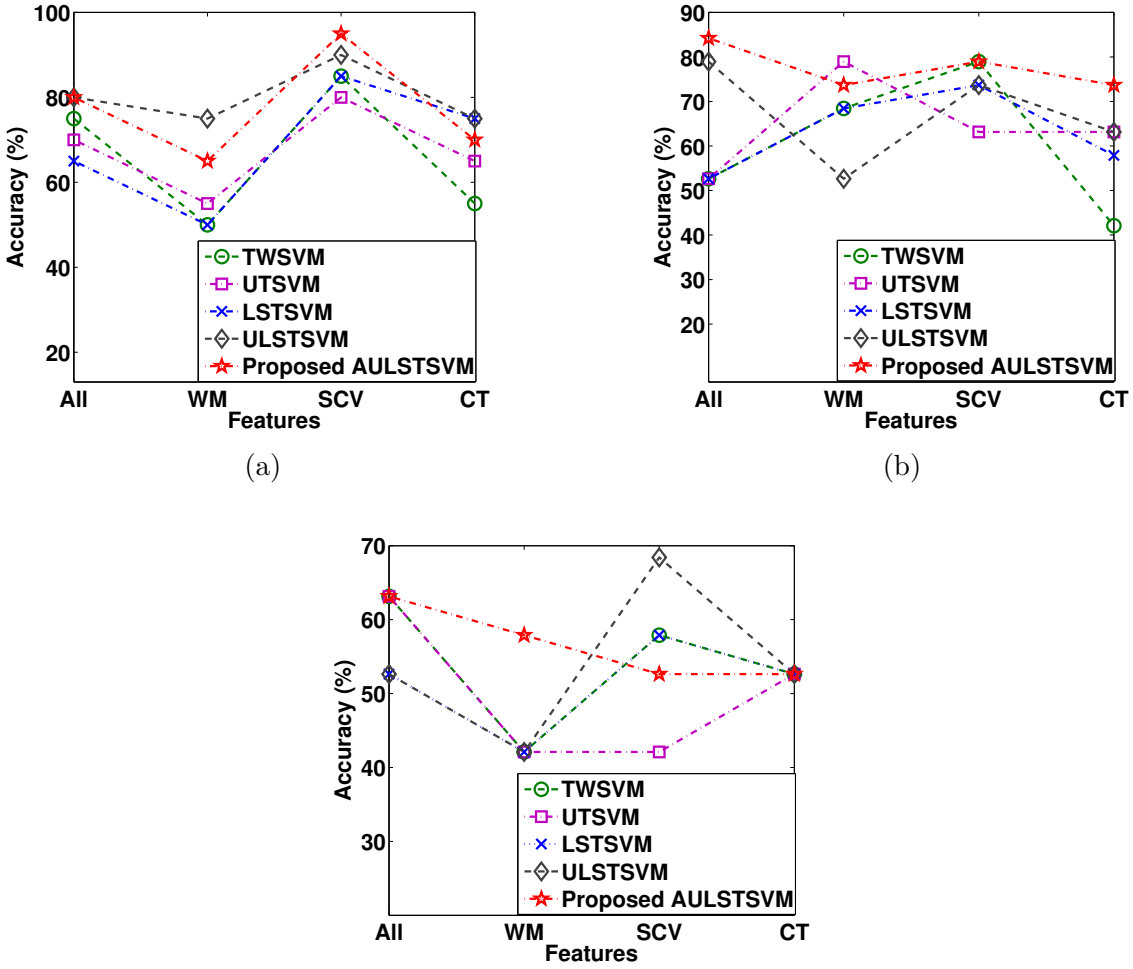


Figure 6.6: Plot showing classification performance for (a) CN vs AD, (b) CN vs MCI, and (c) MCI vs AD by TWSVM, UTSVM, LSTSVM, ULSTSVM, and the proposed AULSTSVM algorithm.

6.2 Proposed universum least squares twin parametric-margin support vector machine (ULSTPMSVM)

In this section, we present a novel parametric-margin based algorithm with universum data for classification problems. The proposed algorithm i.e., ULSTPMSVM involves the solution of system of linear equations making it efficient w.r.t. computation time. The formulations of proposed ULSTPMSVM for the linear and non-linear

cases are discussed in the following subsections.

6.2.1 Linear ULSTPMSVM

The optimization problems of linear ULSTPMSVM are written as

$$\begin{aligned}\min_{w_1,b_1,\eta_1,\psi_1} \quad & \frac{1}{2}(\|w_1\|^2 + b_1^2) + \nu_1 e_2^T(X_2 w_1 + e_2 b_1) + \frac{c_1}{2}\eta_1^T\eta_1 + \frac{c_u}{2}\psi_1^T\psi_1 \\ s.t. \quad & X_1 w_1 + e_1 b_1 = \eta_1, \\ & U w_1 + e_u b_1 + (1-\epsilon)e_u = \psi_1, \end{aligned} \tag{6.27}$$

$$\begin{aligned}\min_{w_2,b_2,\eta_2,\psi_2} \quad & \frac{1}{2}(\|w_2\|^2 + b_2^2) - \nu_2 e_1^T(X_1 w_2 + e_1 b_2) + \frac{c_2}{2}\eta_2^T\eta_2 + \frac{c_u}{2}\psi_2^T\psi_2 \\ s.t. \quad & X_2 w_2 + e_2 b_2 = \eta_2, \\ & U w_2 + e_u b_2 - (1-\epsilon)e_u = \psi_2, \end{aligned} \tag{6.28}$$

where $c_i, i = 1, 2, c_u$ are positive parameters, and $\eta_i, \psi_i, i = 1, 2$ are slack variables.

Using the constraints of Eqs. (6.27) and (6.28) in their respective objective functions, we get

$$\begin{aligned}\min_{w_1,b_1} \quad & \frac{c_1}{2}(\|X_1 w_1 + e_1 b_1\|^2) + \nu_1 e_2^T(X_2 w_1 + e_2 b_1) + \frac{1}{2}(\|w_1\|^2 + b_1^2) \\ & + \frac{c_u}{2}(\|U w_1 + e_u b_1 + (1-\epsilon)e_u\|^2), \end{aligned} \tag{6.29}$$

$$\begin{aligned}\min_{w_2,b_2} \quad & \frac{c_2}{2}(\|X_2 w_2 + e_2 b_2\|^2) - \nu_2 e_1^T(X_1 w_2 + e_1 b_2) + \frac{1}{2}(\|w_2\|^2 + b_2^2) \\ & + \frac{c_u}{2}(\|U w_2 + e_u b_2 - (1-\epsilon)e_u\|^2). \end{aligned} \tag{6.30}$$

Now, taking the gradient of QPP (6.29) w.r.t. w_1 and b_1 and equating to 0, we get

$$c_1 X_1^T(X_1 w_1 + e_1 b_1) + \nu_1 X_2^T e_2 + w_1 + c_u U^T(U w_1 + e_u b_1 + (1-\epsilon)e_u) = 0, \tag{6.31}$$

$$c_1 e_1^T(X_1 w_1 + e_1 b_1) + \nu_1 e_2^T e_2 + b_1 + c_u e_u^T(U w_1 + e_u b_1 + (1-\epsilon)e_u) = 0. \tag{6.32}$$

Combining Eqs. (6.31) and (6.32) and solving, we get

$$[w_1 \;\; b_1]^T = -(c_1 H^T H + c_u O^T O + I)^{-1}(\nu_1 G^T e_2 + (1-\epsilon)c_u O^T e_u), \tag{6.33}$$

where $H = [A; \;\; e_1]$, $G = [B; \;\; e_2]$, and $O = [U; \;\; e_u]$.

Similarly, using Eq. (6.30), we get

$$[w_2 \;\; b_2]^T = (c_2 G^T G + c_u O^T O + I)^{-1}(\nu_2 H^T e_1 + (1-\epsilon)c_u O^T e_u). \tag{6.34}$$

The decision function of linear ULSTPMSVM is same as in Eq. (2.11).

6.2.2 Non-linear ULSTPMSVM

The formulation of non-linear ULSTPMSVM involves kernel generated surfaces. The optimization problem comprises the following two QPPs,

$$\begin{aligned} \min_{w_1,b_1,\eta_1,\psi_1} \quad & \frac{1}{2}(\|w_1\|^2 + b_1^2) + \nu_1 e_2^T (K(X_2, D^T)w_1 + e_2 b_1) + \frac{c_1}{2}\eta_1^T \eta_1 + \frac{c_u}{2}\psi_1^T \psi_1 \\ s.t. \quad & K(X_1, D^T)w_1 + e_1 b_1 = \eta_1, \\ & K(U, D^T)w_1 + e_u b_1 + (1-\epsilon)e_u = \psi_1, \end{aligned} \tag{6.35}$$

$$\begin{aligned} \min_{w_2,b_2,\eta_2,\psi_2} \quad & \frac{1}{2}(\|w_2\|^2 + b_2^2) - \nu_2 e_1^T (K(X_1, D^T)w_2 + e_1 b_2) + \frac{c_2}{2}\eta_2^T \eta_2 + \frac{c_u}{2}\psi_2^T \psi_2 \\ s.t. \quad & K(X_2, D^T)w_2 + e_2 b_2 = \eta_2, \\ & K(U, D^T)w_2 + e_u b_2 - (1-\epsilon)e_u = \psi_2, \end{aligned} \tag{6.36}$$

where $c_i, i = 1, 2, c_u$ are positive parameters, $K(., D^T)$ is the kernel matrix, $D = [A; B]$, and $\eta_i, \psi_i, i = 1, 2$ represent the slack variables.

Substituting the constraints of Eqs. (6.35) and (6.36) in their respective objective functions, we get

$$\min_{w_1,b_1} \quad \frac{1}{2}(\|w_1\|^2 + b_1^2) + \frac{c_1}{2}(\|K(X_1, D^T)w_1 + e_1 b_1\|^2) + \nu_1 e_2^T (K(X_2, D^T)w_1 + e_2 b_1) + \frac{c_u}{2}(\|K(U, D^T)w_1 + e_u b_1 + (1-\epsilon)e_u\|^2), \tag{6.37}$$

$$\min_{w_2,b_2} \quad \frac{1}{2}(\|w_2\|^2 + b_2^2) + \frac{c_2}{2}(\|K(X_2, D^T)w_2 + e_2 b_2\|^2) - \nu_2 e_1^T (K(X_1, D^T)w_2 + e_1 b_2) + \frac{c_u}{2}(\|K(U, D^T)w_2 + e_u b_2 - (1-\epsilon)e_u\|^2). \tag{6.38}$$

Solving similar to the linear case, we get

$$[w_1 \;\; b_1]^T = -(c_1 P^T P + c_u R^T R + I)^{-1}(\nu_1 Q^T e_2 + (1-\epsilon)c_u R^T e_u), \tag{6.39}$$

where $P = [K(X_1, D^T); \;\; e_1]$, $Q = [K(X_2, D^T); \;\; e_2]$, and $R = [K(U, D^T); \;\; e_u]$.

Similarly, using Eq. (6.38), we get

$$[w_2 \;\; b_2]^T = (c_2 Q^T Q + c_u R^T R + I)^{-1}(\nu_2 P^T e_1 + (1-\epsilon)c_u R^T e_u). \tag{6.40}$$

The decision function of non-linear ULSTPMSVM is same as in Eq. (2.10).

6.2.3 Time complexity

The time complexity of proposed ULSTPMSVM is lesser than existing algorithms such as TWSVM and ULSTSVM. In comparison to TWSVM where QPPs are solved, the proposed ULSTPMSVM solves a system of linear equations, leading to lesser computation cost [227]. Moreover, in comparison to ULSTSVM, time complexity of proposed ULSTPMSVM is lesser because ULSTSVM involves an additional matrix multiplication term in its solution (Eqs. 2.36 and 2.37), as compared to proposed ULSTPMSVM in Eqs. (6.33) and (6.34). However, the computation cost of the proposed ULSTPMSVM is higher than LSTSVM. This is due to the incorporation of universum data points in the proposed ULSTPMSVM algorithm.

6.2.4 Experimental results

In this section, we show the results of experiments carried out on real world datasets, along with an application on Alzheimer's disease.

6.2.4.1 Parameter settings

For experiments on real world datasets, 50% data is used for training. The parameters c and ν are chosen from the set $\{10^{-5}, 10^{-4}, ..., 10^{5}\}$, while μ is chosen from $\{2^{-5}, 2^{-4}, ..., 2^{5}\}$. The value for ϵ is selected from $\{0.1, 0.4, 0.7\}$. The universum data is generated by performing random averaging of data points in all the cases [9, 20, 27].

In case of Alzheimer's disease dataset, 40% of the samples are used for training. Freesurfer's recon-all pipeline (version 6.0.1) [205] is applied for processing the structural MRI (sMRI) images as in subsection 4.2.4.

6.2.4.2 Real world datasets

In Table 6.5, performance comparison of the proposed ULSTPMSVM is shown with existing algorithms on 18 real world benchmark datasets. The existing algorithms used for comparison in this work are TWSVM [12], LSTSVM [18], LSTPMSVM [227], and ULSTSVM [71]. One can observe in Table 6.5 that the proposed ULSTPMSVM is showing lowest average rank on the basis of accuracy. In terms of training time, the time taken by proposed ULSTPMSVM is comparable or lesser than the existing algorithms. It is noticeable that the training time of TWSVM is the highest. This is because the solution of TWSVM involves a pair of QPPs, as compared to systems of linear equations in least squares based algorithms.

6.2.4.3 Statistical analysis

To check the statistical difference, the Friedman test [172] is performed with the corresponding posthoc test. First, we assume that there is no difference between the methods. Now, the χ_F^2 value is calculated for Friedman test using average ranks r_i from Table 6.5 as 20.8605.

Table 6.5: Comparison of the proposed ULSTPMSVM with existing algorithms on classification of real world datasets using RBF kernel.

Dataset (Size)	**TWSVM** Accuracy (%) (c, μ) Time (s)	**LSTSVM** Accuracy (%) (c, μ) Time (s)	**LSTPMSVM** Accuracy (%) (c, ν, μ) Time (s)	**ULSTSVM** Accuracy (%) (c_1, c_2, ϵ, μ) Time (s)	**Proposed ULSTPMSVM** Accuracy (%) (c, ν, ϵ, μ) Time (s)
Ecoli-0-1_vs_5 (242 × 6)	94.2149 (10^{-5}, 2^5) 0.0442	90.9091 (10^{-3}, 2^5) 0.0033	95.0413 (10^{-2}, 10^1, 2^4) 0.0035	95.0413 (10^{-5}, 10^{-1}, 0.1, 2^4) 0.0035	**97.5207** (10^1, 10^2, 0.1, 2^5) 0.0035
Ecoli-0-1-4-7_vs_5-6 (334 × 6)	97.006 (10^{-3}, 2^5) 0.0135	94.6108 (10^{-1}, 2^5) 0.0016	97.6048 (10^0, 10^1, 2^4) 0.0018	**98.8024** (10^0, 10^{-3}, 0.7, 2^5) 0.0017	**98.8024** (10^1, 10^{-2}, 0.1, 2^5) 0.0016
Ecoli-0-2-6-7_vs_3-5 (226 × 7)	93.8053 (10^{-3}, 2^5) 0.0079	**97.3451** (10^2, 2^5) 0.0007	96.4602 (10^2, 10^2, 2^5) 0.0007	92.9204 (10^{-1}, 10^{-5}, 0.4, 2^5) 0.0009	96.4602 (10^{-1}, 10^{-2}, 0.1, 2^4) 0.0007
Ecoli-0-3-4-6_vs_5 (206 × 7)	97.0874 (10^{-5}, 2^5) 0.0071	**98.0583** (10^4, 2^5) 0.0006	95.1456 (10^5, 10^2, 2^5) 0.0009	**98.0583** (10^{-1}, 10^{-5}, 0.7, 2^5) 0.0007	**98.0583** (10^1, 10^{-1}, 0.7, 2^5) 0.0007
Ecoli-0-6-7_vs_3-5 (224 × 7)	91.0714 (10^{-3}, 2^4) 0.0106	94.6429 (10^1, 2^5) 0.0007	91.9643 (10^0, 10^1, 2^4) 0.0008	91.0714 (10^{-2}, 10^{-3}, 0.7, 2^5) 0.0008	**95.5357** (10^1, 10^{-1}, 0.1, 2^5) 0.0009
Ecoli4 (336 × 7)	97.6331 (10^{-2}, 2^2) 0.0133	97.6331 (10^{-1}, 2^4) 0.0013	97.6331 (10^3, 10^5, 2^1) 0.0015	**98.2249** (10^{-2}, 10^{-5}, 0.1, 2^3) 0.0015	**98.2249** (10^5, 10^3, 0.4, 2^1) 0.0019
Glass-0-1-6_vs_2 (194 × 9)	89.6907 (10^{-4}, 2^1) 0.0064	91.7526 (10^{-3}, 2^3) 0.0008	**92.7835** (10^5, 10^2, 2^{-1}) 0.0006	87.6289 (10^{-3}, 10^{-5}, 0.4, 2^2) 0.0006	92.7835 (10^{-5}, 10^{-3}, 0.4, 2^{-2}) 0.0006
Glass-0-4_vs_5 (92 × 9)	91.4894 (10^{-1}, 2^4) 0.0058	**100** (10^{-1}, 2^3) 0.0003	95.7447 (10^4, 10^5, 2^4) 0.0005	**100** (10^{-1}, 10^{-5}, 0.1, 2^2) 0.0002	97.8723 (10^3, 10^0, 0.4, 2^0) 0.0002
Heart-stat (270 × 13)	66.9118 (10^0, 2^5) 0.0077	62.5 (10^{-5}, 2^5) 0.0013	65.4412 (10^0, 10^1, 2^5) 0.001	63.2353 (10^1, 10^2, 0.4, 2^5) 0.0009	**67.6471** (10^{-1}, 10^{-2}, 0.7, 2^5) 0.001
Led7digit-0-2-4-5-6-7-8-9_vs_1 (444 × 7)	91.8919 (10^{-4}, 2^1) 0.0236	93.6937 (10^0, 2^2) 0.0024	**96.8468** (10^{-2}, 10^{-1}, 2^0) 0.0026	92.3423 (10^{-1}, 10^{-5}, 0.1, 2^2) 0.0026	92.7928 (10^2, 10^2, 0.1, 2^1) 0.0026
Ecoli-0-1-4-6_vs_5 (282 × 6)	98.5816 (10^{-3}, 2^5) 0.0103	98.5816 (10^3, 2^5) 0.001	98.5816 (10^1, 10^1, 2^4) 0.001	98.5816 (10^{-2}, 10^{-3}, 0.1, 2^5) 0.001	**99.2908** (10^0, 10^{-2}, 0.1, 2^5) 0.001
Ecoli2 (336 × 7)	91.716 (10^{-1}, 2^0) 0.0116	86.9822 (10^{-2}, 2^0) 0.0014	90.5325 (10^{-2}, 10^{-1}, 2^{-3}) 0.0016	**94.0828** (10^0, 10^1, 0.1, 2^{-2}) 0.0013	92.3077 (10^1, 10^{-2}, 0.4, 2^{-1}) 0.0018
Glass4 (282 × 6)	94.4444 (10^{-5}, 2^0) 0.0084	96.2963 (10^{-2}, 2^1) 0.001	94.4444 (10^0, 10^2, 2^{-1}) 0.0007	96.2963 (10^{-2}, 10^{-5}, 0.7, 2^1) 0.0007	**97.2222** (10^2, 10^{-1}, 0.7, 2^1) 0.0008
Breast cancer Wisconsin (683 × 9)	98.538 (10^{-4}, 2^3) 0.032	98.2456 (10^1, 2^4) 0.0064	98.538 (10^{-1}, 10^0, 2^4) 0.0066	98.2456 (10^{-4}, 10^{-4}, 0.4, 2^3) 0.0069	**98.8304** (10^3, 10^3, 0.1, 2^5) 0.0068

Table 6.5 (contd.)

Dataset (Size)	**TWSVM** Accuracy (%) (c, μ) Time (s)	**LSTSVM** Accuracy (%) (c, μ) Time (s)	**LSTPMSVM** Accuracy (%) (c, ν, μ) Time (s)	**ULSTSVM** Accuracy (%) (c_1, c_2, ϵ, μ) Time (s)	**Proposed ULSTPMSVM** Accuracy (%) (c, ν, ϵ, μ) Time (s)
Ecoli3 (336 × 7)	91.716 (10^{-1}, 2^{0}) 0.012	86.9822 (10^{-2}, 2^{0}) 0.0016	90.5325 (10^{-2}, 10^{-1}, 2^{-3}) 0.0021	**94.0828** (10^{0}, 10^{1}, 0.1, 2^{-2}) 0.0016	92.3077 (10^{1}, 10^{-2}, 0.4, 2^{-1}) 0.0015
Yeast1vs7 (460 × 8)	91.7391 (10^{-1}, 2^{-2}) 0.0219	93.0435 (10^{0}, 2^{-1}) 0.003	93.4783 (10^{-4}, 10^{-2}, 2^{-1}) 0.0032	93.4783 (10^{-2}, 10^{-3}, 0.4, 2^{-2}) 0.0031	**93.913** (10^{1}, 10^{-1}, 0.1, 2^{-1}) 0.0042
Ecoli0137vs26 (312 × 7)	95.5128 (10^{-2}, 2^{-2}) 0.0108	96.1538 (10^{-1}, 2^{-1}) 0.0015	**97.4359** (10^{-5}, 10^{-4}, 2^{-2}) 0.0013	96.1538 (10^{1}, 10^{3}, 0.4, 2^{-1}) 0.0013	96.7949 (10^{-1}, 10^{-2}, 0.4, 2^{-1}) 0.0012
Votes (436 × 16)	94.4954 (10^{-1}, 2^{5}) 0.0141	94.4954 (10^{0}, 2^{4}) 0.0023	94.4954 (10^{3}, 10^{2}, 2^{4}) 0.0025	94.0367 (10^{1}, 10^{-4}, 0.7, 2^{4}) 0.0025	**94.9541** (10^{1}, 10^{0}, 0.4, 2^{2}) 0.0025
Average accuracy	92.6414	92.8848	93.4836	93.4602	**94.5177**
Average rank	3.8889	3.4167	3.0278	3.0556	**1.6111**

The F_F value is calculated as

$$F_F = \frac{(18-1)(20.8605)}{18 \times (5-1) - 20.8605} = 6.9345.$$

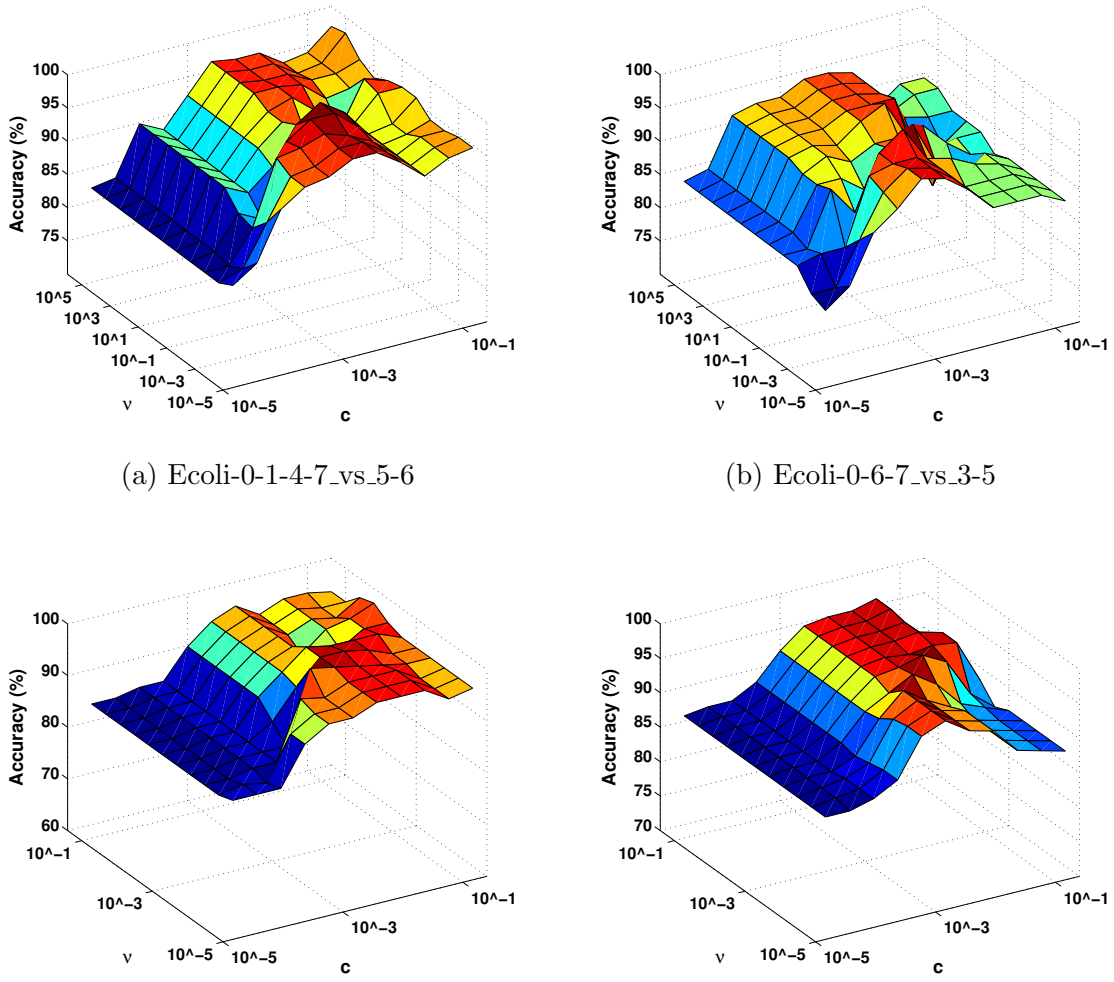
This F-distribution involves $\big(5-1, (5-1)(18-1)\big) = (4, 68)$ degrees of freedom. Thus, for the significance level at $\alpha = 0.05$, the critical value for $F(4, 68)$ is 2.5066. Since $F_F = 6.9345 > 2.5066$, the null hypothesis is rejected.

To check the pairwise difference between the proposed ULSTPMSVM and existing algorithms, we perform the Nemenyi posthoc test [172]. For significant pairwise difference between the methods at significance level of $\alpha = 0.10$, the average ranks of the algorithms shown in Table 6.5 should differ by atleast $2.459\sqrt{\frac{5(5+1)}{6\times 18}} = 1.296$. The pairwise difference between the methods is shown in Table 6.6.

Table 6.6: Pairwise significant difference between the proposed ULSTPMSVM and existing algorithms.

Statistical difference	**TWSVM**	**LSTSVM**	**LSTPMSVM**	**ULSTSVM**
Proposed ULSTPMSVM	**Yes**	**Yes**	**Yes**	**Yes**

It can be stated from Table 6.6 that the proposed ULSTPMSVM is significantly better than TWSVM, LSTSVM, LSTPMSVM, and ULSTSVM algorithms.



(a) Ecoli-0-1-4-7_vs_5-6

(b) Ecoli-0-6-7_vs_3-5

(c) Glass4

(d) Votes

Figure 6.7: Plots showing insensitivity performance for the penalty parameter $c = c_1 = c_2$ with $\nu = \nu_1 = \nu_2$ for proposed ULSTPMSVM using RBF kernel.

6.2.4.4 Insensitivity performance

The insensitivity analysis of proposed ULSTPMSVM is presented in Fig. 6.7. The variation of accuracy w.r.t. the penalty parameter c and ν is shown for 4 datasets viz. Ecoli-0-1-4-7_vs_5-6, Ecoli-0-6-7_vs_3-5, Glass4, and Votes.

One can observe that the accuracy of proposed ULSTPMSVM increases with higher value of c, while ν does not have much effect on the accuracy. However, the accuracy is slightly higher for larger values of ν.

6.2.4.5 Alzheimer's disease classification

In Alzheimer disease data, we have considered three classes namely control normal (CN), Alzheimer's disease (AD), and mild cognitive impairment (MCI) [2, 28]. We include 50 sMRI images of CN and AD each, and 49 sMRI images of MCI, since one MCI image failed to process. The performance of proposed ULSTPMSVM and existing algorithms on classification of Alzheimer data is shown in Table 6.7.

Table 6.7: Performance comparison of proposed ULSTPMSVM on classification of Alzheimer's data.

Dataset	**TWSVM** Accuracy (%)	**LSTSVM** Accuracy (%)	**LSTPMSVM** Accuracy (%)	**ULSTSVM** Accuracy (%)	**Proposed ULSTPMSVM** Accuracy (%)
CN vs AD	**85**	80	80	**85**	76.6667
CN vs MCI	74.5763	59.322	**76.2712**	74.5763	**76.2712**
MCI vs AD	61.0169	44.0678	61.0169	42.3729	**64.4068**

One can see that the proposed ULSTPMSVM performed better than other algorithms in 2 out of 3 datasets i.e., CN vs MCI, and MCI vs AD. The highest accuracy of proposed ULSTPMSVM in MCI vs AD indicates that it may be used for the early diagnosis of Alzheimer's disease. Moreover, the proposed ULSTPMSVM can be used for other diseases such as epilepsy, where the universum data is selected from the dataset itself [20, 26]. This may lead to higher classification accuracy for such problems.

In the next section, we present an improved version of the efficient UTSVM algorithm by introducing regularization in the formulation.

6.3 Proposed improved universum twin support vector machine (IUTSVM)

In this section, we propose an improved universum twin support vector machine (IUTSVM). The SRM principle is implemented by regularization in the primal problems to propose a robust algorithm involving universum data. Moreover, the proposed IUTSVM implicitly makes the matrices non-singular in the optimization problem. The following subsections describe the formulations of the proposed IUTSVM algorithm

in the linear and non-linear forms.

6.3.1 Linear IUTSVM

The linear IUTSVM algorithm comprises the following pair of QPPs:

$$\begin{aligned}
\min_{w_1, b_1, \xi_1, \eta_1} \ & \frac{1}{2}c_3(\|w_1\|^2 + b_1^2) + \frac{1}{2}\|X_1 w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_1 + c_u e_u^T \eta_1, \\
s.t. \quad & -(X_2 w_1 + e_2 b_1) + \xi_1 \geq e_2, \\
& (U w_1 + e_u b_1) + \eta_1 \geq (-1+\epsilon) e_u, \\
& \xi_1 \geq 0, \quad \eta_1 \geq 0
\end{aligned} \tag{6.41}$$

$$\begin{aligned}
\min_{w_2, b_2, \xi_2, \eta_2} \ & \frac{1}{2}c_4(\|w_2\|^2 + b_2^2) + \frac{1}{2}\|X_2 w_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_2 + c_u e_u^T \eta_2, \\
s.t. \quad & (X_1 w_2 + e_1 b_2) + \xi_2 \geq e_1, \\
& -(U w_2 + e_u b_2) + \eta_2 \geq (-1+\epsilon) e_u, \\
& \xi_2 \geq 0, \quad \eta_2 \geq 0,
\end{aligned} \tag{6.42}$$

where $c_i (i = 1, 2, 3, 4)$ and c_u are positive real parameters; ξ_i, $\eta_i (i = 1, 2)$ are slack variables; and $e_i (i = 1, 2)$, e_u are vectors of ones of suitable dimensions.

The Lagrangian of problems (6.41) and (6.42) are written as:

$$\begin{aligned}
L_1 = & \frac{1}{2}c_3(\|w_1\|^2 + b_1^2) + \frac{1}{2}\|X_1 w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_1 + c_u e_u^T \eta_1 \\
& + \alpha_1^T (X_2 w_1 + e_2 b_1) - \xi_1 + e_2) - \beta_1^T \xi_1 \\
& - \mu_1^T ((U w_1 + e_u b_1) + \eta_1 + e_u(1-\epsilon)) - \gamma_1^T \eta_1,
\end{aligned} \tag{6.43}$$

$$\begin{aligned}
L_2 = & \frac{1}{2}c_4(\|w_2\|^2 + b_2^2) + \frac{1}{2}\|X_2 w_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_2 + c_u e_u^T \eta_2 \\
& + \alpha_2^T (-(X_1 w_2 + e_1 b_2) - \xi_2 + e_1) - \beta_2^T \xi_2 \\
& + \mu_2^T ((U w_2 + e_u b_2) - \eta_1 - e_u(1-\epsilon)) - \gamma_2^T \eta_2,
\end{aligned} \tag{6.44}$$

where α_i, β_i, μ_i and γ_i $(i = 1, 2)$ are the Lagrange multipliers.

Using the K.K.T. necessary and sufficient conditions on Eqs. (6.43) and (6.44), the Wolfe duals are obtained as

$$\begin{aligned} &\max_{\alpha_1,\,\mu_1} \; e_2^T\alpha_1 - \frac{1}{2}(\alpha_1^T Q - \mu_1^T R)(P^T P + c_3 I)^{-1}(Q^T\alpha_1 - R^T\mu_1) + (\epsilon - 1)e_u^T\mu_1, \\ &s.t. \;\; 0 \le \alpha_1 \le c_1, \quad 0 \le \mu_1 \le c_u, \end{aligned} \tag{6.45}$$

$$\begin{aligned} &\max_{\alpha_2,\,\mu_2} \; e_2^T\alpha_2 - \frac{1}{2}(\alpha_2^T P - \mu_2^T R)(Q^T Q + c_4 I)^{-1}(P^T\alpha_2 - R^T\mu_2) + (\epsilon - 1)e_u^T\mu_2, \\ &s.t. \;\; 0 \le \alpha_2 \le c_2, \quad 0 \le \mu_2 \le c_u, \end{aligned} \tag{6.46}$$

where $P = [X_1\, e_1]$, $Q = [X_2\, e_2]$, $R = [U\, e_u]$ and I is an identity matrix of appropriate dimension.

The non-parallel hyperplanes $x^T w_1 + b_1 = 0$ and $x^T w_2 + b_2 = 0$ are obtained by using the parameters w_i and b_i, $i = 1, 2$ from the following Eqs. (6.47) and (6.48),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(P^T P + c_3 I)^{-1}(Q^T\alpha_1 - R^T\mu_1), \tag{6.47}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (Q^T Q + c_4 I)^{-1}(P^T\alpha_2 - R^T\mu_2). \tag{6.48}$$

It can be observed that both the matrices $(P^T P + c_3 I)$ and $(Q^T Q + c_4 I)$ are positive definite, and hence the solution of IUTSVM is more robust and stable than that of UTSVM. A new data point is classified to a class using Eq. (2.11).

6.3.2 Non-linear IUTSVM

In non-linear IUTSVM, we consider the following pair of minimization problems:

$$\begin{aligned}\min_{w_1,\, b_1,\, \xi_1,\, \eta_1} \ & \frac{1}{2}c_3(\|w_1\|^2 + b_1^2) + \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_1 + c_u e_u^T \eta_1 \\ s.t. \quad & -(K(X_2, D^T)w_1 + e_2 b_1) + \xi_1 \geq e_2 \\ & (K(U, D^T)w_1 + e_u b_1) + \eta_1 \geq (-1+\epsilon)e_u \\ & \xi_1 \geq 0, \quad \eta_1 \geq 0, \end{aligned} \tag{6.49}$$

$$\begin{aligned}\min_{w_2,\, b_2,\, \xi_2,\, \eta_2} \ & \frac{1}{2}c_4(\|w_2\|^2 + b_2^2) + \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_2 + c_u e_u^T \eta_2 \\ s.t. \quad & (K(X_1, D^T)w_2 + e_1 b_2) + \xi_2 \geq e_1 \\ & -(K(U, D^T)w_2 + e_u b_2) + \eta_2 \geq (-1+\epsilon)e_u \\ & \xi_2 \geq 0, \quad \eta_2 \geq 0, \end{aligned} \tag{6.50}$$

where $c_i (i = 1, 2, 3, 4)$ and c_u are positive real parameters; ξ_i, $\eta_i (i = 1, 2)$ are slack variables and $e_i (i = 1, 2)$ and e_u are the vectors of ones of suitable dimensions.

The Lagrangian of problems (6.49) and (6.50) are written as:

$$\begin{aligned} L_1 = & \frac{1}{2}c_3(\|w_1\|^2 + b_1^2) + \frac{1}{2}\|K(X_1, D^T)w_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_1 + c_u e_u^T \eta_1 \\ & + \alpha_1^T((K(X_2, D^T)w_1 + e_2 b_1) - \xi_1 + e_2) - \beta_1^T \xi_1 \\ & - \mu_1^T((K(U, D^T)w_1 + e_u b_1) + \eta_1 + e_u(1-\epsilon)) - \gamma_1^T \eta_1, \end{aligned} \tag{6.51}$$

$$\begin{aligned} L_2 = & \frac{1}{2}c_4(\|w_2\|^2 + b_2^2) + \frac{1}{2}\|K(X_2, D^T)w_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_2 + c_u e_u^T \eta_2 \\ & + \alpha_2^T(-(K(X_1, D^T)w_2 + e_1 b_2) - \xi_2 + e_1) - \beta_2^T \xi_2 \\ & + \mu_2^T((K(U, D^T)w_2 + e_u b_2) - \eta_1 - e_u(1-\epsilon)) - \gamma_2^T \eta_2. \end{aligned} \tag{6.52}$$

Applying the K.K.T. necessary and sufficient conditions on Eqs. (6.53) and (6.54),

the Wolfe duals are written as

$$\max_{\alpha_1,\mu_1} \; e_2^T\alpha_1 - \frac{1}{2}(\alpha_1^T N - \mu_1^T O)(M^T M + c_3 I)^{-1}(N^T\alpha_1 - O^T\mu_1) + (\epsilon - 1)e_u^T\mu_1$$
$$s.t. \;\; 0 \le \alpha_1 \le c_1, \quad 0 \le \mu_1 \le c_u, \tag{6.53}$$

$$\max_{\alpha_2,\mu_2} \; e_2^T\alpha_2 - \frac{1}{2}(\alpha_2^T M - \mu_2^T O)(N^T N + c_4 I)^{-1}(M^T\alpha_2 - O^T\mu_2) + (\epsilon - 1)e_u^T\mu_2$$
$$s.t. \;\; 0 \le \alpha_2 \le c_2, \quad 0 \le \mu_2 \le c_u, \tag{6.54}$$

where $M = [K(X_1, D^T)\, e_1]$, $N = [K(X_2, D^T)\, e_2]$, $O = [K(U, D^T)\, e_u]$, and I is an identity matrix of appropriate dimension.

The classifying hyperplanes $K(x^T, D^T)w_1 + b_1 = 0$ and $K(x^T, D^T)w_2 + b_2 = 0$ are constructed using the values of the parameters w_i and b_i, $i = 1, 2$ from the following Eqs. (6.55) and (6.56),

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(M^T M + c_3 I)^{-1}(N^T\alpha_1 - O^T\mu_1), \tag{6.55}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (N^T N + c_4 I)^{-1}(M^T\alpha_2 - O^T\mu_2). \tag{6.56}$$

It is visible from Eqs. (6.55) and (6.56) that both $(M^T M + c_3 I)$ and $(N^T N + c_4 I)$ are positive definite due to the regularization term, leading to a more stable solution. A new data point is classified using Eq. (2.10).

6.3.3 Experimental results

The comparison of the proposed IUTSVM is performed with USVM, TWSVM and UTSVM. The RBF kernel is used in all the algorithms. We selected the values of the parameters c, $c_1 = c_2 = c_u$, and $c_3 = c_4$ from the set $\{10^{-5}, \ldots, 10^5\}$ for the different

algorithms. The number of universum samples i.e., r is taken as 10% of the training data for the existing algorithms i.e., USVM, UTSVM and proposed IUTSVM. The value of ϵ is chosen from the set $\{0.1, 0.3, 0.5, 0.6\}$. Universum data is selected from the set obtained by random averaging [27] of data points of the binary classes. The parameter μ is calculated as used in [195] for all the methods.

Table 6.8 shows the performance of the proposed and related algorithms in terms of classification accuracy and training time. The corresponding average ranks are also shown based on accuracy. It can be observed from Table 6.8 that the proposed IUTSVM is having least rank among all the methods. It shows the supremacy of the proposed IUTSVM in comparison to the compared algorithms. The algorithms USVM, UTSVM and proposed IUTSVM take more training time in comparison to TWSVM. This is due to the additional data points of the universum data, which can be traded for the classification accuracy.

To check the significant difference between the methods we performed the Friedman test [172] using Table 6.8. For the Friedman test, we take the null hypothesis that there is no significant difference between the algorithms. Under the null hypothesis, the Friedman statistics is distributed according to $\mathcal{X}_F^2$ with $(k-1)$ degree of freedom and N methods. The χ^2 value is found to be 8.3161. The F_F value is calculated as

$$F_F = \frac{(18-1) \times 8.3161}{18 \times (4-1) - 8.3161} = 3.0946,$$

where F_F is calculated for $(3, 3 \times 17) = (3, 51)$ degrees of freedom on 4 methods and 18 datasets. The critical value for of $F(3, 51)$ at $\alpha = 0.05$ level of significance is 2.786. Here, we reject the null hypothesis since the value of $F_F = 3.0946 > 2.786$. So, there is significant difference between the algorithms. In the Nemenyi posthoc test, the CD is found to be 0.9859. So, according to Table 6.8, there is pairwise significant difference between TWSVM, UTSVM and the proposed IUTSVM.

The insensitivity performance of the proposed IUTSVM for the parameters $c_3 = c_4$ and ϵ is shown in Fig. 6.8. The variation in accuracy is illustrated with changing values of the parameters for datasets i.e., Votes, Bupa or liver-disorders, Ionosphere,

Table 6.8: Performance comparison of the proposed IUTSVM with USVM, TWSVM and UTSVM using RBF kernel.

Dataset (Train size, Test size)	**USVM** Accuracy (%) (c, μ, ϵ) Time(s)	**TWSVM** Accuracy (%) (c_1, μ) Time(s)	**UTSVM** Accuracy (%) (c_1, μ, ϵ) Time(s)	**Proposed IUTSVM** Accuracy (%) $(c_1, c_3, \mu, \epsilon)$ Time(s)
Bupa or liver-disorders $(240 \times 6, 105 \times 6)$	69.5238 $(10^1, 66.2, 0.3)$ 0.8337	70.4762 $(10^{-1}, 66.2)$ 0.049	68.5714 $(10^1, 66.2, 0.6)$ 0.0548	**73.3333** $(10^{-1}, 10^{-5}, 66.2, 0.5)$ 0.0529
Cleveland $(150 \times 13, 147 \times 13)$	80.9524 $(10^0, 5.26, 0.1)$ 0.3216	75.5102 $(10^{-5}, 5.26)$ 0.0109	73.4694 $(10^{-2}, 5.26, 0.5)$ 0.0103	**82.3129** $(10^0, 10^1, 5.26, 0.5)$ 0.0099
Ionosphere $(150 \times 33, 201 \times 33)$	90.0498 $(10^1, 4.39, 0.5)$ 0.3278	92.5373 $(10^{-2}, 4.39)$ 0.0083	92.5373 $(10^{-2}, 4.39, 0.6)$ 0.0095	**93.5323** $(10^{-1}, 10^{-2}, 4.39, 0.3)$ 0.0093
Votes $(200 \times 16, 235 \times 16)$	95.3191 $(10^0, 5.31, 0.5)$ 0.5761	**95.7447** $(10^{-2}, 5.31)$ 0.0125	**95.7447** $(10^{-2}, 5.31, 0.5)$ 0.018	**95.7447** $(10^{-2}, 10^{-4}, 5.31, 0.5)$ 0.0192
Breast cancer wisconsin $(350 \times 9, 333 \times 9)$	**99.0991** $(10^1, 12.53, 0.1)$ 1.7746	98.4985 $(10^{-4}, 12.53)$ 0.0312	98.7988 $(10^{-3}, 12.53, 0.1)$ 0.0406	98.7988 $(10^{-4}, 10^{-5}, 12.53, 0.1)$ 0.0391
Heart-stat $(180 \times 13, 90 \times 13)$	77.7778 $(10^2, 85.98, 0.3)$ 0.4637	**81.1111** $(10^{-1}, 85.98)$ 0.0104	**81.1111** $(10^0, 85.98, 0.1)$ 0.012	80 $(10^0, 10^{-3}, 85.98, 0.5)$ 0.0117
Ndc1k $(400 \times 32, 700 \times 32)$	**91.7143** $(10^3, 571.16, 0.1)$ 2.338	90.1429 $(10^{-1}, 571.16)$ 0.0462	91 $(10^{-1}, 571.16, 0.6)$ 0.058	91.5714 $(10^{-1}, 10^{-5}, 571.16, 0.6)$ 0.059
Pima $(350 \times 8, 418 \times 8)$	78.2297 $(10^3, 2.23, 0.3)$ 1.7374	75.1196 $(10^0, 2.23)$ 0.0329	77.512 $(10^0, 2.23, 0.6)$ 0.04	**79.4258** $(10^0, 10^{-5}, 2.23, 0.3)$ 0.0403
Splice $(1000 \times 60, 2175 \times 60)$	**90.023** $(10^1, 11.98, 0.6)$ 15.2782	89.6552 $(10^{-1}, 11.98)$ 0.4038	89.1494 $(10^{-1}, 11.98, 0.6)$ 0.528	89.7011 $(10^1, 10^{-2}, 11.98, 0.5)$ 0.5272
Wdbc $(250 \times 30, 319 \times 30)$	92.79 $(10^5, 944.41, 0.3)$ 0.943	**95.9248** $(10^1, 944.41)$ 0.0239	95.6113 $(10^0, 944.41, 0.3)$ 0.0276	93.1034 $(10^{-2}, 10^{-5}, 944.41, 0.6)$ 0.0275
Australian-credit $(540 \times 14, 150 \times 14)$	**90.6667** $(10^{-1}, 5.31, 0.3)$ 4.3106	88 $(10^{-1}, 5.31)$ 0.0883	88 $(10^{-1}, 5.31, 0.6)$ 0.1159	90 $(10^{-1}, 10^{-2}, 5.31, 0.6)$ 0.1105
Yeast3 $(500 \times 8, 984 \times 8)$	93.1911 $(10^3, 0.41, 0.3)$ 3.7007	**94.4106** $(10^{-2}, 0.41)$ 0.1377	**94.4106** $(10^{-1}, 0.41, 0.6)$ 0.1561	93.8008 $(10^4, 10^4, 0.41, 0.6)$ 0.1708

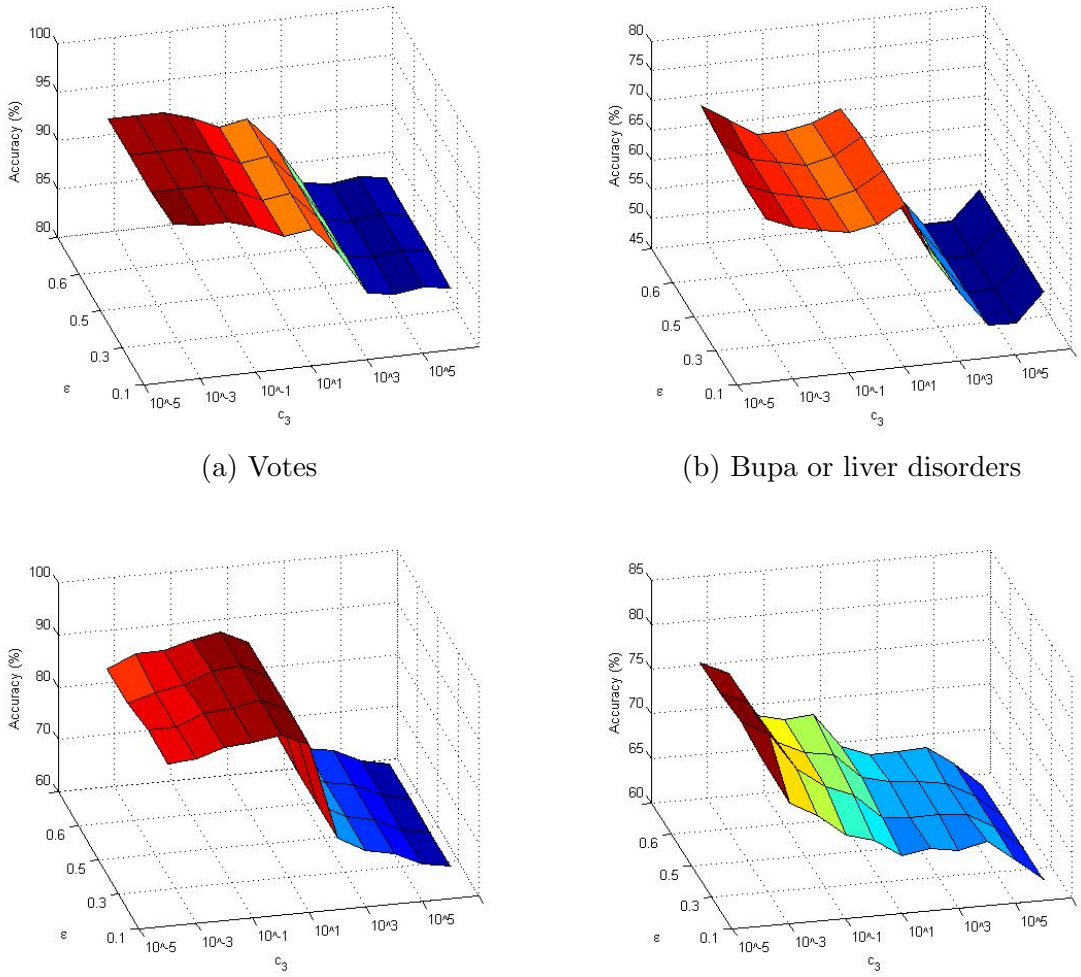
Table 6.8 (contd.)

Dataset (Train size, Test size)	**USVM** Accuracy (%) (c, μ, ϵ) Time(s)	**TWSVM** Accuracy (%) (c_1, μ) Time(s)	**UTSVM** Accuracy (%) (c_1, μ, ϵ) Time(s)	**Proposed IUTSVM** Accuracy (%) $(c_1, c_3, \mu, \epsilon)$ Time(s)
Ecoli-0-1_vs_2-3-5 $(120 \times 7, 124 \times 7)$	**97.5806** $(10^1, 67.42, 0.5)$ 0.2116	94.3548 $(10^{-1}, 67.42)$ 0.008	93.5484 $(10^{-2}, 67.42, 0.5)$ 0.0091	95.1613 $(10^0, 10^{-1}, 67.42, 0.1)$ 0.0093
Ecoli-0-1-4-7_vs_5-6 $(150 \times 6, 182 \times 6)$	97.2527 $(10^1, 67.54, 0.3)$ 0.3281	98.3516 $(10^{-2}, 67.54)$ 0.0116	98.3516 $(10^{-1}, 67.54, 0.1)$ 0.0125	**98.9011** $(10^{-1}, 10^{-3}, 67.54, 0.1)$ 0.0123
Ecoli-0-2-6-7_vs_3-5 $(110 \times 7, 114 \times 7)$	94.7368 $(10^2, 59.17, 0.1)$ 0.1779	92.9825 $(10^{-1}, 59.17)$ 0.007	93.8596 $(10^{-2}, 59.17, 0.1)$ 0.0081	**95.614** $(10^{-5}, 10^{-3}, 59.17, 0.1)$ 0.0074
Ecoli-0-3-4-6_vs_5 $(100 \times 7, 105 \times 7)$	**98.0952** $(10^0, 65.84, 0.3)$ 0.1482	94.2857 $(10^{-4}, 65.84)$ 0.0068	**98.0952** $(10^{-1}, 65.84, 0.1)$ 0.0074	**98.0952** $(10^1, 10^0, 65.84, 0.3)$ 0.0073
Ecoli-0-6-7_vs_3-5 $(110 \times 7, 112 \times 7)$	**97.3214** $(10^1, 57.95, 0.1)$ 0.178	93.75 $(10^{-2}, 57.95)$ 0.0073	91.9643 $(10^{-1}, 57.95, 0.6)$ 0.0086	96.4286 $(10^{-1}, 10^{-3}, 57.95, 0.1)$ 0.0082
Yeast-2_vs_4 $(250 \times 8, 264 \times 8)$	95.8333 $(10^1, 0.35, 0.1)$ 0.9188	95.0758 $(10^{-1}, 0.35)$ 0.0239	96.5909 $(10^{-1}, 0.35, 0.3)$ 0.0294	**97.7273** $(10^{-1}, 10^{-2}, 0.35, 0.3)$ 0.0298
Average accuracy	90.5643	89.774	89.907	**91.2918**
Average rank	2.4444	2.9167	2.8333	**1.8056**

and Pima-Indians with accuracies as 95.74%, 73.33%, 93.53%, and 79.43%. For these datasets, IUTSVM is performing better for lower values of c_3, which is also reflected in Table 6.8. This shows that small values of the regularization parameter c_3 are more suitable for the proposed IUTSVM on these datasets. However, the parameter ϵ does not show much effect on the performance of IUTSVM in terms of accuracy.

6.4 Summary

In this chapter, we presented three novel algorithms based on universum based twin SVM algorithms. The proposed algorithms incorporate some important improvements over the existing formulations. First, we presented a novel universum



(a) Votes

(b) Bupa or liver disorders

(c) Ionosphere

(d) Pima Indians

Figure 6.8: Insensitivity performance of the proposed IUTSVM to the parameters $c_3 = c_4$ with ϵ using RBF kernel.

based algorithm, termed as efficient angle based universum least squares twin support vector machine (AULSTSVM). The proposed AULSTSVM removes the drawback of existing least squares based algorithms w.r.t. computation time. Moreover, the proposed AULSTSVM gives better generalization performance due to the incorporation of prior information in the training. Instead of the traditional twin hyperplane based approach for obtaining the decision function, the proposed AULSTSVM needs only one hyperplane for classification.

We also proposed a novel universum based algorithm in this chapter termed as universum least squares twin parametric-margin support vector machine (UL-

STPMSVM). The formulation of ULSTPMSVM is an alternative approach towards universum based learning. Lastly, we presented an improved universum twin support vector machine. The proposed algorithm introduces the SRM principle in the formulation of UTSVM. There is no ill-conditioning of the matrices in calculating the inverse in the proposed IUTSVM. The proposed algorithms show high generalization performance with lesser training time in comparison to existing algorithms. Moreover, the proposed AULSTSVM, and ULSTPMSVM performed well for classification of Alzheimer's disease, showing their applicability on real world biomedical applications.

In the next chapter, we discuss the paradigm of unsupervised learning using SVM. We present a novel formulation for twin SVM based clustering using a projection based approach.

Chapter 7

Projection based twin support vector clustering

In this chapter, we present an unsupervised learning algorithm using a twin SVM based approach. Clustering is a prominent unsupervised learning technique. In the literature, various plane based clustering algorithms are proposed, such as the twin support vector clustering (TWSVC) algorithm. In this work, we propose an alternative algorithm based on projection axes termed as least squares projection twin support vector clustering (LSPTSVC)[1]. The proposed LSPTSVC finds projection axis for every cluster in a manner that minimizes the within class scatter, and keeps the clusters of other classes far away. The following sections discuss the proposed algorithm both theoretically as well as experimentally.

7.1 Proposed algorithm

In this section, we present the formulations of proposed least squares projection twin support vector clustering (LSPTSVC) with detailed analysis of the experimental results. The idea of proposed scheme is illustrated in Fig. 7.1, where a projection axis is generated to cluster the data points. The proposed LSPTSVC minimizes the

[1]**B. Richhariya**, M. Tanveer, Alzheimer's Disease Neuroimaging Initiative. Least squares projection twin support vector clustering (LSPTSVC). *Information Sciences*, Elsevier, 533:1-23, 2020, DOI: `https://doi.org/10.1016/j.ins.2020.05.001`.
[**SCI Indexed Impact Factor: 6.795**]

scatter of a cluster, while keeping the data points of other clusters far away. We also include the regularization term in the objective function to control the structural risk of the model [5]. The regularization term also helps in avoiding the ill-conditioning of the matrices for calculating the inverse [167].

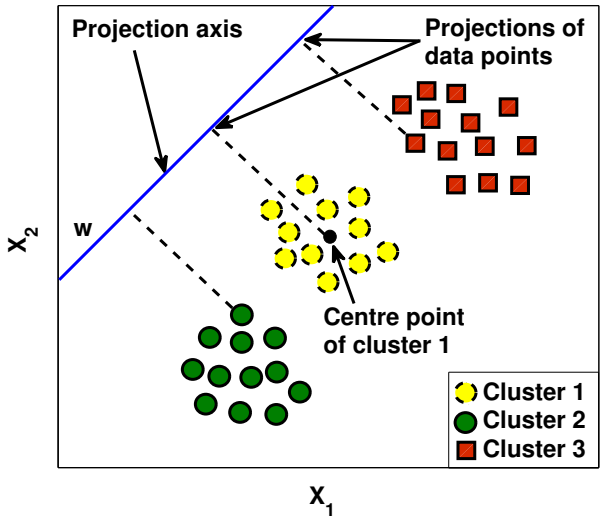


Figure 7.1: Clustering by proposed LSPTSVC.

7.1.1 Linear LSPTSVC

The optimization problem of linear LSPTSVC is described as

$$\begin{aligned}
\min_{w_i^{j+1}} \ & \frac{1}{2}\sum_{p=1}^{m_i}\left((w_i^{j+1})^T x_p^{(i)} - (w_i^{j+1})^T \frac{1}{m_i}\sum_{p=1}^{m_i} x_p^{(i)}\right)^2 + \frac{c_1}{2}\sum_{q=1}^{\overline{m}_i}(\xi_{iq}^{j+1})^2 + \frac{c_2}{2}\|w_i^{j+1}\|^2 \\
s.t. \ & \left|(w_i^{j+1})^T \overline{x}_q^{(i)} - (w_i^{j+1})^T \frac{1}{m_i}\sum_{p=1}^{m_i} x_p^{(i)}\right| + \xi_{iq}^{j+1} = 1, \\
& q = 0, 1, \ldots, \overline{m}_i, \quad i = 0, 1, \ldots, N,
\end{aligned} \tag{7.1}$$

where $c_1, c_2 > 0$ are parameters, w_i^{j+1} represents the weight vector of $(j+1)^{th}$ iteration, $j = 0, 1, \ldots$, and the slack variable is represented by ξ_{iq}^{j+1}. The data points of a cluster, and rest of the clusters are represented by $x_p^{(i)}$ and $\overline{x}_q^{(i)}$ respectively. Here, N is the number of clusters, and $\overline{m}_i = (m - m_i)$.

The QPP (7.1) is formulated by setting an objective function that minimizes intra class variance, while maximizing the inter class distance using the constraints. To solve this optimization problem, we use the concave-convex (CCCP) procedure [228]. Thus, the objective function of QPP (7.1) can be rewritten as

$$\begin{aligned}\min_{w_i^{j+1}} \ & \frac{1}{2}(w_i^{j+1})^T S_i w_i^{j+1} + \frac{c_1}{2}\sum_{q=1}^{\overline{m}_i}(\xi_{iq}^{j+1})^2 + \frac{c_2}{2}\|w_i^{j+1}\|^2 \\ s.t. \ & T\Big(\Big|\overline{X}_i w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^{j+1}\Big|\Big) + \xi_{iq}^{j+1} = \overline{e}_i, \end{aligned} \tag{7.2}$$

where $T(.)$ is the first order Taylor series expansion, e_i and $\overline{e}_i$ represent the vector of ones of size p and q respectively. The matrix S_i is written as

$$S_i = \sum_{p=1}^{m_i}(x_p^{(i)} - s_i)(x_p^{(i)} - s_i)^T, \tag{7.3}$$

where $s_i = \frac{1}{m_i}\sum_{p=1}^{m_i} x_p^{(i)}$ is the centre point of each cluster. Eq. (7.4) can be rewritten as

$$S_i = (X_i - e_i s_i^T)^T(X_i - e_i s_i^T). \tag{7.4}$$

The QPP (7.2) can be rewritten by substituting the constraints in the objective function as

$$L = \frac{1}{2}(w_i^{j+1})^T S_i w_i^{j+1} + \frac{c_1}{2}\Big\|-T\Big(\Big|\overline{X}_i w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^{j+1}\Big|\Big) + \overline{e}_i\Big\|^2 + \frac{c_2}{2}\|w_i^{j+1}\|^2. \tag{7.5}$$

Now, the value of the Taylor series expansion is written by using the subgradient [168, 169] of $\Big|\overline{X}_i w_i^j - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^j\Big|$ w.r.t. w_i^j as

$$\begin{aligned} T\Big(\Big|\overline{X}_i w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^{j+1}\Big|\Big) = diag\Big(sign\big(\overline{X}_i w_i^j - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^j\big)\Big) \\ \Big(\overline{X}_i w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^{j+1}\Big). \end{aligned} \tag{7.6}$$

Substituting the value of T(.) in Eq. (7.6), we get

$$L = \frac{1}{2}(w_i^{j+1})^T S_i w_i^{j+1} + \frac{c_2}{2}\|w_i^{j+1}\|^2 + \frac{c_1}{2}\Big\|-diag\Big(sign\big(\overline{X}_i w_i^j - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^j\big)\Big)\Big(\overline{X}_i w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^{j+1}\Big) + \overline{e}_i\Big\|^2. \tag{7.7}$$

Solving the gradient of Eq. (7.7) w.r.t. w_i^{j+1} and equating to 0, we get

$$S_i w_i^{j+1} + c_2 w_i^{j+1} + c_1 G_i^T\big(G_i w_i^{j+1} - \overline{e}_i\big) = 0,$$

where

$$G_i = diag\Big(sign(\overline{X}_i w_i^j - \frac{1}{m_i}\overline{e}_i e_i^T X_i w_i^j\Big)\Big(\overline{X}_i - \frac{1}{m_i}\overline{e}_i e_i^T X_i\Big). \tag{7.8}$$

Solving Eq. (7.8) for w_i^{j+1}, we get

$$w_i^{j+1} = \Big(G_i^T G_i + \frac{S_i}{c_1} + \frac{c_2}{c_1} I_i\Big)^{-1} G_i^T \overline{e}_i. \tag{7.9}$$

For a testing sample x_t, the label y is determined by the following formula:

$$y(x_t) = arg \min_{i=1,2,\ldots,N} \Big|w_i^T x_t - \frac{1}{m_i} e_i^T X_i w_i\Big|. \tag{7.10}$$

For the initialization of the labels, nearest neighbour graph (NNG) [166] algorithm is used in LSPTSVC. The algorithm for linear LSPTSVC is shown in Alg. 7.1.

Algorithm 7.1 Linear LSPTSVC

1: ***Inputs:***
   1.1 Unlabelled data $X \in \mathbb{R}^n$.
2: ***Initialization:***
   2.1 Label assignment: $Y_0 \leftarrow NNG(X)$.
   2.2 Initialize weight vector w_i^0 for each cluster $i = 0, 1, \ldots, N$:
      $w_i^0 = Eigenvector(S_i)$, for smallest eigenvalue of S_i.
3: ***CCCP process:***
   3.1 For each cluster i, calculate w_i^{j+1} for $j = 0$ using the initial weight vector w_i^0.
      3.1.1 $w_i^{j+1} = LSPTSVC(X, w_i^j)$ using Eq. (7.9).
      3.1.2 **if** $(\|w_i^{j+1} - w_i^j\| > tolerance)$
         $j = j + 1$
         go to step 3.1.1
      3.1.3 **else**
         go to step 4
4: ***Assign cluster labels:***
   4.1 Assign new labels to the data points for each cluster i.
      4.1.1 $Y_{k+1} = Decision\_function(X, w_i)$ using Eq. (7.10), initially with $k = 0$.
      4.1.2 **if** $\big(\|Y_{k+1} - Y_k\| \neq 0\big)$
         $k = k + 1$
         go to step 3
      4.1.3 **else**
         go to step 5
5: ***Output:***
   Return data labels Y, and projection vectors w_i, $i = 0, 1, \ldots, N$.

7.1.2 Non-linear LSPTSVC

The optimization problem of non-linear LSPTSVC is described as

$$\begin{aligned} \min_{w_i^{j+1}} \ & \frac{1}{2}(w_i^{j+1})^T Z_i w_i^{j+1} + \frac{c_1}{2}\sum_{q=1}^{\overline{m}_i}(\xi_{iq}^{j+1})^2 + \frac{c_2}{2}\|w_i^{j+1}\|^2 \\ s.t. \ & T\Big(\Big|K(\overline{X}_i, M^T)w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T K(X_i, M^T)w_i^{j+1}\Big|\Big) + \xi_{iq}^{j+1} = \overline{e}_i, \\ & q = 0, 1, \ldots, \overline{m}_i, \quad i = 0, 1, \ldots, N, \end{aligned} \tag{7.11}$$

where $c_1, c_2 > 0$ are parameters, $j = 0, 1, \ldots,$ and ξ_{iq}^{j+1} represents the slack variable, $T(.)$ is the first order Taylor series expansion, $K(., M^T)$ is the kernel function [159],

and $M = [X_1, X_2, \ldots, X_N]$. The matrix Z_i is written as

$$Z_i = \sum_{p=1}^{m_i} \big(K(x_p^{(i)}, M^T) - z_i\big)\big(K(x_p^{(i)}, M^T) - z_i\big)^T, \tag{7.12}$$

where $z_i = \frac{1}{m_i} \sum_{p=1}^{m_i} K(x_p{}^{(i)}, M^T)$. Now, Eq. (7.12) can be rewritten as

$$Z_i = \big(K(X_i, M^T) - e_i z_i^T\big)^T \big(K(X_i, M^T) - e_i z_i^T\big). \tag{7.13}$$

Now, QPP (7.13) can be written by using the constraints in the objective function as

$$\begin{aligned} L =& \frac{1}{2}(w_i^{j+1})^T Z_i w_i^{j+1} + \frac{c_1}{2}\Big\| -T\Big(\Big|K(\overline{X}_i, M^T)w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T K(X_i, M^T) w_i^{j+1}\Big|\Big) + \overline{e}_i\Big\|^2 \\ &+ \frac{c_2}{2}\|w_i^{j+1}\|^2. \end{aligned} \tag{7.14}$$

Substituting the value of T(.) in (7.14) for the CCCP procedure, we get

$$\begin{aligned} L =& \frac{1}{2}(w_i^{j+1})^T Z_i w_i^{j+1} + \frac{c_2}{2}\|w_i^{j+1}\|^2 + \\ & \frac{c_1}{2}\Big\| -diag\Big(sign\big(K(\overline{X}_i, M^T)w_i^j - \frac{1}{m_i}\overline{e}_i e_i^T K(X_i, M^T) w_i^j\big)\Big) \\ & \Big(K(\overline{X}_i, M^T)w_i^{j+1} - \frac{1}{m_i}\overline{e}_i e_i^T K(X_i, M^T) w_i^{j+1}\Big) + \overline{e}_i\Big\|^2. \end{aligned} \tag{7.15}$$

Now, solving the gradient of (7.15) w.r.t. w_i^{j+1} and equating to 0, we get

$$Z_i w_i^{j+1} + \frac{c_2}{2} w_i^{j+1} + c_1 U^T\big(U w_i^{j+1} - \overline{e}_i\big) = 0,$$

where

$$\begin{aligned} U_i = diag\Big(sign\big(K(\overline{X}_i, M^T)w_i^j - \frac{1}{m_i}\overline{e}_i e_i^T K(X_i, M^T) w_i^j\big)\Big) \\ \Big(K(\overline{X}_i, M^T) - \frac{1}{m_i}\overline{e}_i e_i^T K(X_i, M^T)\Big), \end{aligned} \tag{7.16}$$

Solving Eq. (7.16) for w_i^{j+1}, we get

$$w_i^{j+1} = \Big(U_i^T U_i + \frac{Z_i}{c_1} + \frac{c_2}{c_1} I_i\Big)^{-1} U_i^T \overline{e}_i. \tag{7.17}$$

The above equation involves matrix inversion of order $m \times m$. This leads to high computation time for datasets having very large m as compared to number of features. In order to reduce the computation cost of calculating the inverse, we use the Sherman-Morrison-Woodbury (SMW) formula [40].

We can write Eq. (7.17) as

$$w_i^{j+1} = \Big(U_i^T U_i + \frac{c_2}{c_1} I_i + \frac{D_i^T D_i}{c_1}\Big)^{-1} U_i^T \overline{e}_i, \tag{7.18}$$

where $D = \big(K(X_i, M^T) - e_i z_i^T\big)$. Now, using the SMW formula (Eq. 6.20), we obtain the following expression:

$$w_i^{j+1} = \big(A_i^{-1} - A_i^{-1} U_i^T (I_i + U_i A_i^{-1} U_i^T)^{-1} U_i A_i^{-1}\big) U_i^T \overline{e}_i, \tag{7.19}$$

where $A_i^{-1} = \frac{c_1}{c_2}\big(I_i - D_i^T (c_2 I_i + D_i D_i^T)^{-1} D_i\big)$.

For calculating w_i^{j+1}, instead of computing inverse of size $(m \times m)$, we need to compute one inverse of size $(m_i \times m_i)$, and other of size $(m - m_i) \times (m - m_i)$, $\forall i = 1, 2, \ldots, N$.

For a testing sample x_t, the label y is determined as follows,

$$y(x_t) = arg \min_{i=1,2,\ldots,N} \Big| w_i^T K(x_t, M^T) - \frac{1}{m_i} e_i^T K(X_i, M^T) w_i \Big|. \tag{7.20}$$

The algorithm for non-linear LSPTSVC is shown in Alg. 7.2.

Lemma 7.1.1 *Let $X \in \mathbb{R}^{m \times n}$, $S \in \mathbb{R}^{n \times n}$, $m > n$. Then, $S = (X - es)^T (X - es)$ is a positive semidefinite matrix, where $s = \frac{1}{m} \sum_{p=1}^{m} x_p^T$.*

Proof: Let $T = (X - es)$. Then, $S = T^T T$ is a symmetric matrix.

Algorithm 7.2 Non-linear LSPTSVC

1: ***Inputs:***
1.1 Unlabelled data $X \in \mathbb{R}^n$.
1.2 Kernel matrix K obtained using kernel function.
2: ***Initialization:***
2.1 Label assignment: $Y_0 \leftarrow NNG(K)$.
2.2 Initialize weight vector w_i^0 for each cluster $i = 0, 1, \ldots, N$:
$w_i^0 = Eigenvector(Z_i)$, for smallest eigenvalue of Z_i.
3: ***CCCP process:***
3.1 For each cluster i, calculate w_i^{j+1} for $j = 0$ using the initial weight vector w_i^0.
3.1.1 $w_i^{j+1} = LSPTSVC(K, w_i^j)$ using Eq. (7.17).
3.1.2 **if** $(\|w_i^{j+1} - w_i^j\| > tolerance)$
$j = j + 1$
go to step 3.1.1
3.1.3 **else**
go to step 4
4: ***Assign cluster labels:***
4.1 Assign new labels to the data points for each cluster i.
4.1.1 $Y_{k+1} = Decision\_function(K, w_i)$ using Eq. (7.20), initially with $k = 0$.
4.1.2 **if** $\big(\|Y_{k+1} - Y_k\| \neq 0\big)$
$k = k + 1$
go to step 3
4.1.3 **else**
go to step 5
5: ***Output:***
Return data labels Y, and projection vectors w_i, $i = 0, 1, \ldots, N$.

Now, for any $w \in \mathbb{R}^n$,

$$w^T S w = w^T T^T T w$$
$$\implies w^T S w = \|Tw\|^2 \geq 0. \tag{7.21}$$

Therefore, S is positive semidefinite.

Theorem 7.1.1 *Let $X \in \mathbb{R}^{p \times n}$, $m > n$, and $S = (X - es)^T (X - es)$. Then, the global minimum of:*

$$\min_{w} \; w^T S w$$
$$s.t. \; w^T w = 1, \tag{7.22}$$

is obtained for any eigenvector w of S with minimum eigenvalue. The minimum value of (7.22) is positive iff S is positive definite or equivalently iff $rank(X - es) = n$.

Proof: Firstly, we write the Lagrangian of Eq. (7.22),

$$L = w^T S w - \lambda\big(w^T w - 1\big). \tag{7.23}$$

Now, we use the Karush-Kuhn-Tucker (K.K.T.) optimality conditions,

$$\frac{\partial L}{\partial w} = Sw - \lambda w = 0, \tag{7.24}$$
$$\frac{\partial L}{\partial \lambda} = w^T w - 1 = 0. \tag{7.25}$$

From (7.24) and (7.25), we get

$$\lambda = w^T S w. \tag{7.26}$$

Putting the value of λ in (7.24), we get

$$Sw = (w^T S w) w, \tag{7.27}$$

which is equivalent to

$$Sw = kw, \tag{7.28}$$

where $k = w^T S w$, which is to be minimized in Eq. (7.22). Hence, the smallest eigenvalue of S gives the eigenvector w to achieve global minimum of (7.22) [163].

Remark: If $S \in \mathbb{R}^{n \times n}$ is positive definite, then S is non-singular.

Theorem 7.1.2 *Let $X \in \mathbb{R}^{p \times n}$, $G \in \mathbb{R}^{q \times n}$, $S = (X - es)^T(X - es) \in \mathbb{R}^{n \times n}$, and $I \in \mathbb{R}^{n \times n}$ is the identity matrix. Then, the matrix $\left(G^TG + \frac{S}{c_1} + \frac{c_2}{c_1}I\right)$ is invertible $\forall\, c_1, c_2 > 0$.*

Proof: Here, G^TG and S are positive semidefinite matrices from Lemma 7.1.1, and $\frac{c_2}{c_1}I_i$ is positive definite for $c_1, c_2 > 0$. Also, for any vector $w \in \mathbb{R}^n, w \neq 0$, the sum of a positive semidefinite and positive definite matrix is always positive definite as shown below:

Let A and B be a positive definite and positive semidefinite matrix respectively. Then, for any vector $w \in \mathbb{R}^n, w \neq 0$,

$$w^TAw > 0, \tag{7.29}$$

$$w^TBw \geq 0, \tag{7.30}$$

Adding (7.29) & (7.30), we get

$$(w^TAw + w^TBw) > 0, \tag{7.31}$$

$$w^T(A + B)w > 0. \tag{7.32}$$

Therefore, the square matrix $\left(G^TG + \frac{S_i}{c_1} + \frac{c_2}{c_1}I\right)$ is always non-singular, and thus invertible.

7.1.3 Convergence

The proposed LSPTSVC described in Algs. 7.1 and 7.2 converges in a finite number of steps. This is because CCCP method always finds a local minimum, and thus converges as discussed in [228]. Moreover, in the cluster assignment process, every data point is assigned to closest hyperplane [163]. So, the overall objective function cannot increase. Thus, the algorithm converges based on any of the following terminating conditions:

(i). Same labels assigned to data points in two consecutive iterations.

(ii). Non-decrease in the overall objective function.

7.1.4 Time complexity

In comparison to TWSVC which solves large sized QPPs to solve the clustering problem, the proposed LSPTSVC only needs to solve sets of linear equations. The time complexity of solving the QPP in linear TWSVC is $O(N(\overline{m}_i)^3)$ for $m = m_i + \overline{m}_i$ samples, N classes and $\overline{m}_i$ constraints, $i = 1, 2, \ldots, N$. The complexity of calculating the matrix inverse is about $O(Nn^3)$ [167]. So, the complexity of TWSVC becomes $O(N(\overline{m}_i^3 + n^3))$. In case of non-linear TWSVC, the complexity for QPP is $O(N\overline{m}_i^3)$, where $\overline{m}_i$ is the number of constraints, and for inverse is about $O(Nm^3)$. Therefore, the complexity becomes $O(N(\overline{m}_i^3 + m^3))$. The time complexity of TBSVC is same as TWSVC.

The solution of linear LSPTSVC requires the inversion of N matrices of size $n \times n$. Thus, the time complexity of solving the inverses in Eq. (7.9) is $O(Nn^3)$. In the non-linear case, N matrix inverses of size m_i, and $\overline{m}_i$ need to be calculated. Therefore, the time complexity is $O(N(m_i^3 + \overline{m}_i^3)$, $i = 1, 2, \ldots, N$. The time complexity of LSPTSVC is lower than TWSVC which leads to lesser training time. LSTWSVC has similar time complexity as LSPTSVC, except the fact that it needs to calculate the bias. Consequently, the number of linear equations in LSPTSVC is less than LSTWSVC by one equation.

Moreover, in the initialization process, the proposed LSPTSVC only needs to find w. On the other hand, the bias b is also calculated in case of existing plane based clustering algorithms.

7.1.5 Proposed LSPTSVC vs LSPTSVM

The LSPTSVM [158] is a supervised learning algorithm that performs classification of data points by constructing projection axes for each class. The decision function shown in Eq. (2.45) is constructed in a manner to keep the projected data well

separated. However, LSPTSVM minimizes the within class variance of one class, and keeps the scatter of the other class far away on one side of the axis. This is shown by the constraints of QPP (2.38).

We extended the projection axes based approach for unsupervised learning. The proposed LSPTSVC performs clustering by minimizing the within cluster variance, and keeping the scatter of the other clusters far away on both sides of the axes as shown in Fig. 7.1. This is a result of the constraints of QPP (7.1). Moreover, LSPTSVM solves the optimization problem by system of linear equations, whereas proposed LSPTSVC solves linear equations in multiple iterations of the CCCP procedure to obtain the projection axes.

In contrast to LSPTSVM, the proposed LSPTSVC involves an initialization procedure for the weights. Since the scatter matrix S involved in the projection based algorithms is positive semidefinite, we presented the initialization procedure based on eigenvalue of S. In terms of time complexity, LSPTSVC requires more computation time than LSPTSVM, since it involves the CCCP iterative procedure, and mostly deals with multiclass clustering of data. However, the proposed LSPTSVC is computationally more efficient than LSTWSVC. This is analogous to the lesser computation cost of LSPTSVM in comparison to LSTSVM [158].

7.1.6 Experimental results

In this section, performance of the proposed LSPTSVC is compared with existing techniques on the basis of clustering accuracy and training time. The algorithms used for comparison are FCM [162], kPC [163], TWSVC [166], TBSVC [167], LSTWSVC [168], and FLSTWSVC [168]. Among these, FCM is a distance based technique using fuzzy memberships, while rest are plane based algorithms. We use 13 synthetic and 10 real world benchmark datasets to assess the performance of the proposed model with linear and non-linear kernels. Moreover, the performance of LSPTSVC is compared with existing algorithms on 12 large scale datasets. Performance comparison on real world applications are also presented viz. clustering of faces, facial expressions, and Alzheimer's disease data.

The synthetic benchmark datasets are downloaded from the website (`https://github.com/deric/clustering-benchmark`), while the real world and large scale datasets are taken from UCI repository [170]. For applications, facial images are downloaded from AT&T database (`https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`) of AT&T Laboratories Cambridge, and facial expression data is taken from JAFFE database [229]. For application on Alzheimer's data, the same dataset is used as in 4.2.2, with the same pre-processing pipeline.

7.1.6.1 Parameter settings

In Table 7.1 and 7.2, 50% of total data samples are used for training and rest for testing. The value of the parameters c_1, c_2 are selected as in subsection 3.1.5 for all the cases. The tolerance value for the CCCP process is set as 0.001 in all the algorithms. In FCM, the weighting exponent i.e., m is selected from the set $\{1.25, 1.5, 1.75, 2\}$ [162]. In case of large datasets, the value of c_1, c_2 is fixed as 1 [158], and μ is set as 2^5 for all the algorithms.

In all the existing algorithms except FCM, the initialization of weights is performed using kPC algorithm [163], while LSPTSVC is initialized using Theorem 7.1.1. For initialization of cluster labels, the well known nearest neighbour graph (NNG) technique [166] is used for all the algorithms except FLSTWSVC which uses fuzzy NNG [168]. However, in case of large datasets, a set of randomly generated cluster labels are used for initialization of all the algorithms.

7.1.6.2 Results on benchmark datasets

The comparison of the proposed LSPTSVC with existing methods viz. FCM [162], kPC [163], TWSVC [166], TBSVC [167], LSTWSVC [168], and FLSTWSVC [168] is shown in Table 7.1 for linear case. One can observe that the proposed LSPTSVC is showing better performance w.r.t. clustering accuracy in comparison to existing algorithms. This is also justified by the lowest average rank of LSPTSVC i.e., 2.0217 for all the datasets. Moreover, the training time of LSPTSVC is lesser than TWSVC and TBSVC. This is due to the fact that the proposed LSPTSVC solves a set of

linear equations to obtain the projection axis. In contrast, TWSVC and TBSVC solve computationally expensive QPPs.

In comparison to LSTWSVC, LSPTSVC is slightly faster in most datasets since it only needs to calculate the weight vector w and not the bias b. For FLSTWSVC, the training time is higher than proposed LSPTSVC due to the overhead of calculation of fuzzy membership. The comparison of computation time of kPC and FCM with proposed LSPTSVC is not given, since they are not twin SVM based algorithms. However, we have shown the training time of all the algorithms in the tables.

Table 7.1 also shows the number of mis-clustered [230] data points for every algorithm. The mis-clustered data points are calculated by counting the pair of data points with cluster mismatch. One can observe in Table 7.1 that even in terms of the mis-clustered data points, proposed LSPTSVC obtains the least rank i.e., 2.3696. One can notice that for some datasets, the best performing algorithm in terms of accuracy is not having the least number of mis-clustered data points. This can be attributed to the imbalance in the number of data points of clusters in a dataset.

The Win-Tie-Loss comparison is also shown in Table 7.1. The clustering accuracy of proposed LSPTSVC is compared with existing algorithms in a Win-Tie-Loss scenario for all the datasets. It is evident that LSPTSVC is having a 'Win' scenario for all the compared algorithms. The highest 'Win' case is in comparison to FCM, kPC, and TWSVC algorithm. This is because FCM algorithm is based on distance from neighbouring data points, while the datasets have varying data distributions. Moreover, LSPTSVC initializes its weights using the eigenvectors and then converges, while kPC obtains its hyperplanes as the eigenvectors. In comparison to TWSVC, proposed LSPTSVC involves the concept of within class scatter minimization leading to better clustering accuracy. However, the proposed LSPTSVC is having some losses in case of TBSVC, LSTWSVC, and FLSTWSVC. For more analysis on significance of the proposed algorithm, statistical analysis is presented in section 4.4.

The clusters identified by the proposed and existing algorithms using linear kernel for 3MC synthetic dataset are shown in Fig. 7.2. The actual clusters in the dataset are shown in Fig. 7.2(a). One can easily notice that the clusters labelled by the proposed

Table 7.1: Performance comparison on clustering accuracy (%), number of misclustered data points (# Miss), and training time of the proposed LSPTSVC with existing algorithms using linear kernel. The Win-Tie-Loss calculation is based on accuracy, and 's' represents time in seconds.

Dataset (Size, clusters)	**FCM** [162] Accuracy # Miss - Time (s)	**kPC** [163] Accuracy # Miss - Time (s)	**TWSVC** [166] Accuracy # Miss (c_1) Time (s)	**TBSVC** [167] Accuracy # Miss (c_1, c_2) Time (s)	**LSTWSVC** [168] Accuracy # Miss (c_1, c_2) Time (s)	**FLSTWSVC** [168] Accuracy # Miss (c_1, c_2) Time (s)	**Proposed LSPTSVC** Accuracy # Miss (c_1, c_2) Time (s)
Synthetic							
3MC (400×2, 3)	51.37 89 - 0.0151	66.3 89 - 0.00008	68.96 80 (10^0) 0.0191	90.1 18 ($10^{-5}, 10^2$) 0.0162	90.1 18 ($10^{-5}, 10^2$) 0.00015	89.29 16 ($10^1, 10^1$) 0.0037	**96.02** 6 ($10^2, 10^5$) 0.00009
Aggregation (788×2, 7)	77.64 152 - 0.1394	79.36 189 - 0.00015	79.97 183 (10^{-1}) 0.2811	78.86 178 ($10^0, 10^0$) 0.2869	80.82 187 ($10^{-3}, 10^{-1}$) 0.00324	**89.62** 95 ($10^{-1}, 10^{-1}$) 0.0418	83.55 149 ($10^1, 10^5$) 0.00185
Compound (399×2, 6)	78.41 72 - 0.0718	73.56 97 - 0.0001	79.88 81 (10^0) 0.055	80.14 84 ($10^{-1}, 10^{-4}$) 0.054	79.88 92 ($10^{-3}, 10^{-4}$) 0.00039	79.76 80 ($10^{-2}, 10^{-2}$) 0.0077	**86.61** 51 ($10^{-5}, 10^3$) 0.00026
R15 (600×2, 15)	92.17 126 - 0.0243	91.4 133 - 0.00019	96.18 64 (10^{-2}) 0.3949	93.49 102 ($10^{-1}, 10^{-3}$) 0.4091	95.58 75 ($10^{-3}, 10^{-5}$) 0.00238	**97.71** 36 ($10^0, 10^{-5}$) 0.0486	97.07 48 ($10^0, 10^4$) 0.00123
Zelnik5 (512×2, 4)	68.73 112 - 0.0163	78.73 96 - 0.00011	78.43 99 (10^2) 0.055	**95.08** 15 ($10^1, 10^{-1}$) 0.0472	85.99 58 ($10^{-5}, 10^{-2}$) 0.00044	82.75 72 ($10^{-5}, 10^{-5}$) 0.0107	91.4 26 ($10^2, 10^{-5}$) 0.00022
2d-4c-no9 (876×2, 4)	85.77 114 - 0.0931	77.66 130 - 0.00014	97.49 14 (10^{-2}) 0.1426	96.85 18 ($10^{-1}, 10^1$) 0.1436	97.01 17 ($10^{-1}, 10^1$) 0.00169	96.37 22 ($10^0, 10^{-5}$) 0.0244	**98.59** 8 ($10^{-3}, 10^2$) 0.00084
Longsquare (900×2, 6)	81.67 179 - 0.0114	83.46 159 - 0.00016	84.27 127 (10^{-5}) 0.228	86.47 128 ($10^{-5}, 10^{-1}$) 0.2248	85.78 130 ($10^{-4}, 10^{-1}$) 0.00408	90.15 101 ($10^{-2}, 10^{-4}$) 0.0537	**90.93** 94 ($10^2, 10^4$) 0.00259
Hepta (212×3, 7)	77.05 43 - 0.006	87.57 38 - 0.00011	99.03 2 (10^{-3}) 0.0268	99.03 2 ($10^{-3}, 10^{-5}$) 0.0268	98.49 3 ($10^{-3}, 10^{-5}$) 0.00016	94.9 11 ($10^{-1}, 10^{-2}$) 0.0031	**100** 0 ($10^{-3}, 10^0$) 0.00015
Zelnik3 (266×2, 3)	74.87 31 - 0.0054	81.6 20 - 0.00007	80.25 22 (10^{-5}) 0.0113	80.25 22 ($10^{-5}, 10^{-2}$) 0.0109	80.25 22 ($10^{-2}, 10^{-2}$) 0.00009	75.1 30 ($10^{-5}, 10^{-5}$) 0.0018	**83.96** 17 ($10^2, 10^{-5}$) 0.00006

Table 7.1 (contd.)

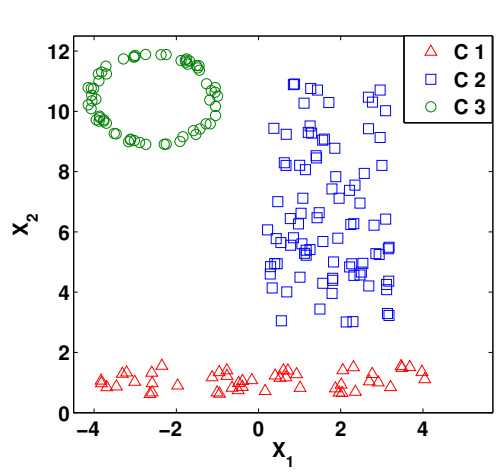
Dataset (Size, clusters)	**FCM** [162] Accuracy # Miss - Time (s)	**kPC** [163] Accuracy # Miss - Time (s)	**TWSVC** [166] Accuracy # Miss (c_1) Time (s)	**TBSVC** [167] Accuracy # Miss (c_1, c_2) Time (s)	**LSTWSVC** [168] Accuracy # Miss (c_1, c_2) Time (s)	**FLSTWSVC** [168] Accuracy # Miss (c_1, c_2) Time (s)	**Proposed LSPTSVC** Accuracy # Miss (c_1, c_2) Time (s)
Pathbased (300×2, 3)	61.71 74 - 0.0156	70.36 48 - 0.00007	71.22 45 (10^{-1}) 0.0135	**71.44** 45 ($10^{-2}, 10^{-1}$) 0.0131	70.7 47 ($10^{-2}, 10^{-1}$) 0.00011	**71.44** 45 ($10^{-5}, 10^{-5}$) 0.0023	71.35 45 ($10^{1}, 10^{5}$) 0.00008
Zelnik1 (399×2, 3)	56.47 70 - 0.022	58.72 71 - 0.00009	56.05 71 (10^{-1}) 0.0136	59.59 72 ($10^{2}, 10^{0}$) 0.0144	56.05 70 ($10^{-2}, 10^{-3}$) 0.00011	49.58 74 ($10^{0}, 10^{-1}$) 0.0031	**59.65** 69 ($10^{2}, 10^{-5}$) 0.00008
Ds2c2sc13 (588×2, 13)	87.44 115 - 0.4318	90.46 103 - 0.00035	93.08 69 (10^{-3}) 0.3133	93.01 70 ($10^{-3}, 10^{-5}$) 0.3154	93.17 72 ($10^{-3}, 10^{-5}$) 0.0021	93.06 79 ($10^{-1}, 10^{-2}$) 0.0437	**93.45** 81 ($10^{1}, 10^{1}$) 0.00128
2d-4c-no4 (863×2, 4)	87.68 66 - 0.0048	62.52 205 - 0.0003	67.13 131 (10^{2}) 0.1697	85.22 70 ($10^{0}, 10^{5}$) 0.0754	85.22 70 ($10^{-5}, 10^{5}$) 0.00261	62.96 205 ($10^{3}, 10^{-3}$) 0.0275	**93.38** 24 ($10^{1}, 10^{5}$) 0.00148
Real world							
Ecoli (336×7, 8)	**71.02** 71 - 0.0144	37.79 81 - 0.00034	63.47 81 (10^{5}) 0.1092	63.44 82 ($10^{3}, 10^{4}$) 0.0563	61.98 80 ($10^{1}, 10^{-5}$) 0.00033	56.19 81 ($10^{2}, 10^{5}$) 0.0083	68.25 84 ($10^{1}, 10^{1}$) 0.00035
Zoo (101×16, 7)	54.53 24 - 0.0181	72.73 15 - 0.0023	82.04 14 (10^{-3}) 0.0309	89.8 11 ($10^{-1}, 10^{0}$) 0.0217	88.98 11 ($10^{-5}, 10^{3}$) 0.00026	**90.12** 11 ($10^{-1}, 10^{1}$) 0.0057	88.41 12 ($10^{2}, 10^{-5}$) 0.00022
Wine (178×13, 3)	34.22 43 - 0.0048	56.03 43 - 0.00016	73.88 25 (10^{-2}) 0.0087	71.71 26 ($10^{-5}, 10^{0}$) 0.009	75.74 21 ($10^{-3}, 10^{-1}$) 0.00011	**79.01** 17 ($10^{1}, 10^{-3}$) 0.0026	74.06 23 ($10^{-1}, 10^{5}$) 0.00009
Iris (150×4, 3)	32.68 37 - 0.0059	62.7 29 - 0.00007	91.53 5 (10^{-5}) 0.0083	93.08 4 ($10^{-1}, 10^{0}$) 0.0083	**94.7** 3 ($10^{-1}, 10^{1}$) 0.00007	**94.7** 3 ($10^{-5}, 10^{0}$) 0.0009	**94.7** 3 ($10^{-5}, 10^{0}$) 0.00006
Seeds (210×7, 3)	32.77 52 - 0.0072	**76.58** 33 - 0.00015	75.66 33 (10^{-2}) 0.0116	75.99 34 ($10^{-5}, 10^{-2}$) 0.0103	75.44 35 ($10^{-5}, 10^{-2}$) 0.00008	75.66 33 ($10^{-1}, 10^{-4}$) 0.0022	74.56 27 ($10^{2}, 10^{2}$) 0.00008
Teachingeval (151×5, 3)	33.98 36 - 0.0047	42.85 37 - 0.00009	48.32 37 (10^{2}) 0.0093	52.65 37 ($10^{-1}, 10^{0}$) 0.0083	51.32 37 ($10^{-2}, 10^{2}$) 0.0001	49.05 37 ($10^{-5}, 10^{1}$) 0.0011	**56.25** 36 ($10^{1}, 10^{-1}$) 0.00006

Table 7.1 (contd.)

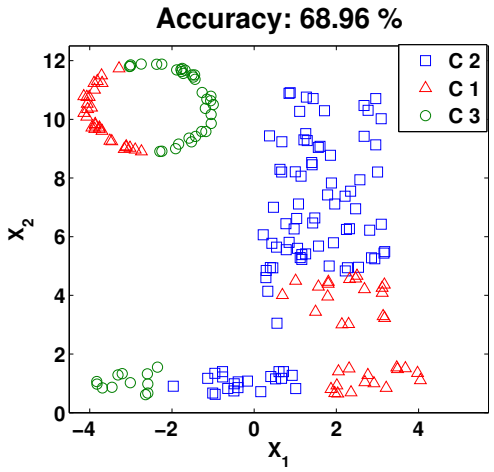
Dataset (Size, clusters)	**FCM** [162] Accuracy # Miss - Time (s)	**kPC** [163] Accuracy # Miss - Time (s)	**TWSVC** [166] Accuracy # Miss (c_1) Time (s)	**TBSVC** [167] Accuracy # Miss (c_1, c_2) Time (s)	**LSTWSVC** [168] Accuracy # Miss (c_1, c_2) Time (s)	**FLSTWSVC** [168] Accuracy # Miss (c_1, c_2) Time (s)	**Proposed LSPTSVC** Accuracy # Miss (c_1, c_2) Time (s)
Tae (150×5, 3)	32.68 37 - 0.0088	45.15 36 - 0.0001	52.61 37 (10^3) 0.0094	52.61 37 ($10^5, 10^{-5}$) 0.0098	49.48 35 ($10^1, 10^3$) 0.00011	32.68 37 ($10^{-5}, 10^{-5}$) 0.0012	**56.61** 36 ($10^2, 10^4$) 0.00006
Hayes-roth (132×4, 3)	**62.75** 27 - 0.0098	57.67 31 - 0.00007	55.99 32 (10^1) 0.0077	55.34 33 ($10^0, 10^{-4}$) 0.0077	53.38 32 ($10^{-2}, 10^{-2}$) 0.00008	52.73 28 ($10^0, 10^{-5}$) 0.0008	53.94 32 ($10^2, 10^3$) 0.00005
Shuttle (1486×9, 5)	63.25 167 - 0.3162	51.55 359 - 0.00235	62.85 251 (10^{-1}) 1.1994	**87.77** 54 ($10^{-2}, 10^2$) 1.4062	78.54 124 ($10^0, 10^5$) 0.00828	82.12 104 ($10^2, 10^4$) 0.1927	78.54 112 ($10^0, 10^4$) 0.00783
Libras (360×90, 15)	70.99 89 - 0.4334	64.14 88 - 0.0491	66.16 89 (10^{-5}) 0.1845	83.27 88 ($10^{-1}, 10^{-1}$) 0.1641	86.72 87 ($10^1, 10^{-4}$) 0.00584	87.65 85 ($10^{-4}, 10^{-3}$) 0.0261	**87.81** 88 ($10^0, 10^{-4}$) 0.00503
Average accuracy	65.23	69.38	75.81	80.25	79.41	77.9	**82.22**
Average rank (Accuracy)	5.8043	5.4783	4.1522	3.1522	3.6304	3.7609	**2.0217**
Average rank (# Miss)	5.1304	5.3696	4.1957	3.9783	3.6304	3.3261	**2.3696**
Win-Tie-Loss	21-0-2	21-0-2	21-0-2	17-0-6	18-2-3	15-1-7	

LSPTSVC in Fig. 7.2(f) are similar to the original clusters in Fig. 7.2(a). This may be attributed to minimization of the within class variance of projected data, while keeping the projected data of other classes at unit distance from centre of the cluster.

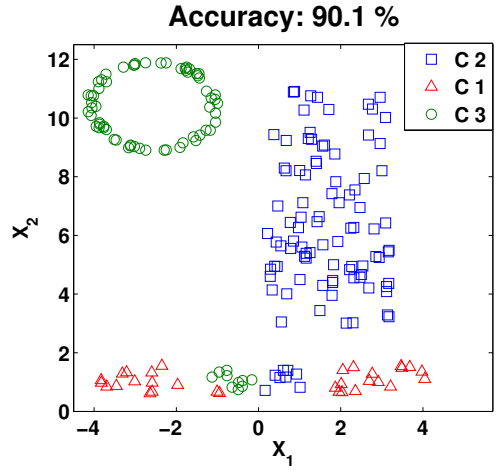
Table 7.2 shows the performance of the proposed LSPTSVC for non-linear case. One can notice that clustering accuracy of the proposed LSPTSVC is better than existing algorithms for most of the datasets. This is also evident from the average rank of LSPTSVC which is the lowest among algorithms in Table 7.2 i.e., 1.7391. Moreover, the proposed non-linear LSPTSVC is having 'Win' situation with all the algorithms in most datasets. This is due to the effect of RBF kernel resulting in non-linear projection axes. Similar to the linear case, the training time of proposed LSPTSVC is also lesser than existing algorithms.



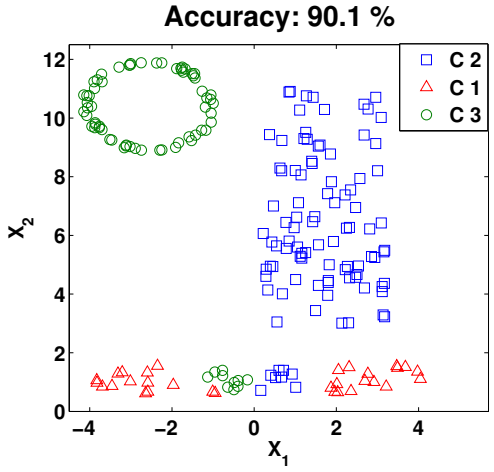
(a) Actual clusters



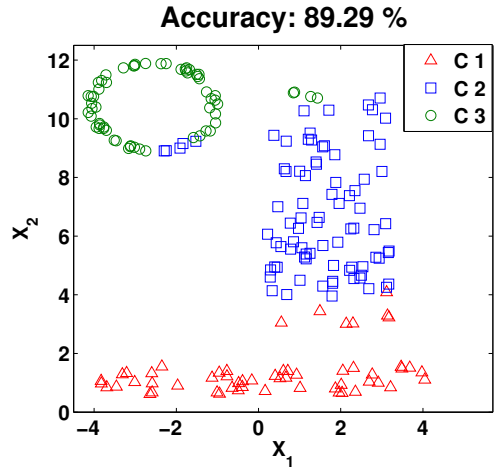
(b) TWSVC



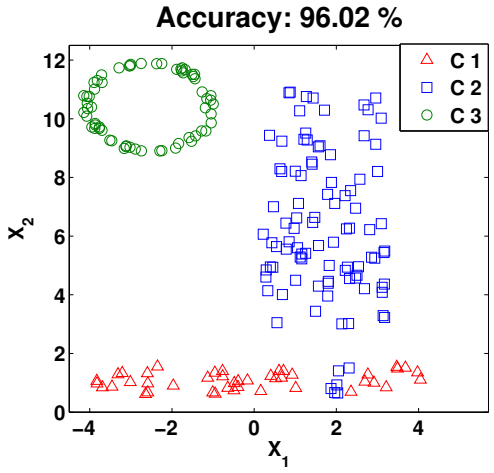
(c) TBSVC



(d) LSTSVC



(e) FLSTSVC



(f) Proposed LSPTSVC

Figure 7.2: Plot showing performance of proposed LSPTSVC in comparison to existing algorithms using linear kernel for 3MC dataset. In the legend 'C' means cluster.

Table 7.2: Performance comparison on clustering accuracy (%), number of misclustered data points (# Miss), and training time of the proposed LSPTSVC with existing algorithms using RBF kernel. The Win-Tie-Loss calculation is based on accuracy, and 's' represents time in seconds.

Dataset (Size, clusters)	**FCM** [162] Accuracy # Miss - Time (s)	**kPC** [163] Accuracy # Miss - Time (s)	**TWSVC** [166] Accuracy # Miss (c_1, μ) Time (s)	**TBSVC** [167] Accuracy # Miss $(c_1 = c_2, \mu)$ Time (s)	**LSTWSVC** [168] Accuracy # Miss (c_1, c_2, μ) Time (s)	**FLSTWSVC** [168] Accuracy # Miss (c_1, c_2, μ) Time (s)	**Proposed LSPTSVC** Accuracy # Miss (c_1, c_2, μ) Time (s)
Synthetic							
3MC (400×2, 3)	51.37 89 - 0.0151	69.18 63 - 0.0383	66.32 97 $(10^0, 2^5)$ 0.0977	60.81 93 $(10^1, 2^0)$ 0.0993	**88.85** 18 $(10^{-3}, 10^{-1}, 2^5)$ 0.045	72.27 62 $(10^3, 10^{-4}, 2^4)$ 0.0455	82.31 29 $(10^1, 10^1, 2^5)$ 0.0444
Aggregation (788×2, 7)	77.64 152 - 0.1394	92.21 77 - 0.638	92.36 77 $(10^{-3}, 2^3)$ 0.8281	92.85 70 $(10^{-5}, 2^1)$ 0.7539	92.8 71 $(10^{-5}, 10^{-2}, 2^1)$ 0.29	**94.97** 42 $(10^{-1}, 10^{-1}, 2^3)$ 0.2958	92.99 70 $(10^{-5}, 10^{-5}, 2^1)$ 0.2835
Compound (399×2, 6)	78.41 72 - 0.0718	88.57 53 - 0.1013	91.34 44 $(10^{-5}, 2^0)$ 0.1506	91.37 41 $(10^{-2}, 2^1)$ 0.1482	**93.24** 38 $(10^{-3}, 10^{-4}, 2^0)$ 0.0557	90.81 43 $(10^{-3}, 10^{-5}, 2^1)$ 0.0588	93.05 40 $(10^{-2}, 10^2, 2^2)$ 0.0527
R15 (600×2, 15)	92.17 126 - 0.0243	99.43 8 - 0.6568	99.73 4 $(10^{-5}, 2^5)$ 0.6917	**99.8** 3 $(10^{-5}, 2^4)$ 0.7201	99.42 7 $(10^{-5}, 10^{-4}, 2^{-1})$ 0.2268	99.43 7 $(10^{-5}, 10^{-5}, 2^{-1})$ 0.23	99.57 5 $(10^{-5}, 10^{-4}, 2^{-1})$ 0.2197
Zelnik5 (512×2, 4)	68.73 112 - 0.0163	75.96 87 - 0.0946	82.07 76 $(10^{-3}, 2^5)$ 0.1574	**86.85** 53 $(10^{-2}, 2^1)$ 0.1356	84.03 80 $(10^{-1}, 10^{-2}, 2^1)$ 0.0837	85.64 67 $(10^{-5}, 10^{-5}, 2^2)$ 0.0876	84.86 61 $(10^2, 10^5, 2^{-3})$ 0.0794
2d-4c-no9 (876×2, 4)	85.77 114 - 0.0931	98.66 8 - 0.3521	98.39 7 $(10^{-5}, 2^2)$ 0.6216	97.81 10 $(10^{-5}, 2^1)$ 0.6215	97.07 15 $(10^{-2}, 10^1, 2^2)$ 0.2801	**99.16** 5 $(10^{-5}, 10^{-4}, 2^1)$ 0.2892	98.28 8 $(10^{-4}, 10^{-2}, 2^2)$ 0.2798
Longsquare (900×2, 6)	81.67 179 - 0.0114	89.27 107 - 0.5345	64.72 200 $(10^{-2}, 2^1)$ 1.0026	78.86 192 $(10^0, 2^2)$ 0.8856	81.08 198 $(10^{-4}, 10^2, 2^2)$ 0.3905	93.22 65 $(10^{-3}, 10^{-4}, 2^2)$ 0.3796	**93.76** 49 $(10^0, 10^4, 2^2)$ 0.3779
Hepta (212×3, 7)	77.05 43 - 0.006	**100** 0 - 0.018	**100** 0 $(10^{-5}, 2^1)$ 0.0563	**100** 0 $(10^{-5}, 2^1)$ 0.0553	**100** 0 $(10^{-5}, 10^{-5}, 2^3)$ 0.0156	**100** 0 $(10^{-5}, 10^{-5}, 2^0)$ 0.0176	**100** 0 $(10^{-5}, 10^{-5}, 2^1)$ 0.0144
Zelnik3 (266×2, 3)	74.87 31 - 0.0054	83.12 19 - 0.0253	83.39 18 $(10^{-1}, 2^5)$ 0.048	81.7 20 $(10^{-4}, 2^2)$ 0.0455	84.51 17 $(10^1, 10^4, 2^{-4})$ 0.0206	**100** 0 $(10^{-5}, 10^{-3}, 2^{-5})$ 0.0214	**100** 0 $(10^0, 10^3, 2^{-4})$ 0.0192
Pathbased (300×2, 3)	61.71 74 - 0.0156	63.9 64 - 0.0252	57.49 74 $(10^4, 2^5)$ 0.0638	68.21 60 $(10^0, 2^5)$ 0.0571	70.56 46 $(10^{-5}, 10^0, 2^5)$ 0.0256	75.7 34 $(10^{-2}, 10^{-5}, 2^2)$ 0.026	**82.26** 23 $(10^1, 10^3, 2^2)$ 0.0255

Table 7.2 (contd.)

Dataset (Size, clusters)	**FCM** [162] Accuracy # Miss - Time (s)	**kPC** [163] Accuracy # Miss - Time (s)	**TWSVC** [166] Accuracy # Miss (c_1, μ) Time (s)	**TBSVC** [167] Accuracy # Miss $(c_1 = c_2, \mu)$ Time (s)	**LSTWSVC** [168] Accuracy # Miss (c_1, c_2, μ) Time (s)	**FLSTWSVC** [168] Accuracy # Miss (c_1, c_2, μ) Time (s)	**Proposed LSPTSVC** Accuracy # Miss (c_1, c_2, μ) Time (s)
	56.47	52.6	59.82	58.52	68.57	73.41	**74.16**
Zelnik1	70	74	69	69	57	53	56
(399×2, 3)	-	-	$(10^{-2}, 2^4)$	$(10^{-2}, 2^1)$	$(10^2, 10^0, 2^{-4})$	$(10^5, 10^3, 2^{-4})$	$(10^4, 10^1, 2^2)$
	0.022	0.0217	0.0433	0.0381	0.0268	0.0275	0.0248
	87.44	84.25	94.28	94.78	**95.14**	93.47	95
Ds2c2sc13	115	114	45	43	41	77	41
(588×2, 13)	-	-	$(10^{-5}, 2^{-3})$	$(10^{-3}, 2^{-4})$	$(10^{-5}, 10^{-5}, 2^{-4})$	$(10^{-3}, 10^{-4}, 2^{-3})$	$(10^{-5}, 10^{-4}, 2^{-5})$
	0.4318	0.3848	0.5756	0.5546	0.1956	0.2095	0.1902
	87.68	71.55	65.39	64.93	62.14	89.65	**95.27**
2d-4c-no4	66	101	155	156	162	66	20
(863×2, 4)	-	-	$(10^{-4}, 2^1)$	$(10^0, 2^2)$	$(10^{-5}, 10^5, 2^2)$	$(10^{-1}, 10^3, 2^1)$	$(10^1, 10^5, 2^2)$
	0.0048	0.3495	0.5077	0.4275	0.2884	0.2889	0.2773
Real world							
	71.02	68.31	61.13	60.61	63.45	55.01	**73.28**
Ecoli	71	69	84	83	74	67	81
(336×7, 8)	-	-	$(10^{-5}, 2^3)$	$(10^4, 2^3)$	$(10^1, 10^{-4}, 2^5)$	$(10^{-1}, 10^0, 2^5)$	$(10^1, 10^3, 2^5)$
	0.0144	0.0496	0.1331	0.1175	0.0434	0.0466	0.0407
	54.53	81.71	86.53	86.86	85.88	80.33	**89.31**
Zoo	24	16	11	14	13	15	9
(101×16, 7)	-	-	$(10^{-5}, 2^3)$	$(10^{-2}, 2^2)$	$(10^3, 10^{-5}, 2^3)$	$(10^{-1}, 10^{-5}, 2^2)$	$(10^1, 10^{-1}, 2^3)$
	0.0181	0.0039	0.0379	0.0311	0.0039	0.0078	0.0035
	34.22	65.27	68.69	72.32	74.26	59.24	**75.18**
Wine	43	30	27	22	23	38	21
(178×13, 3)	-	-	$(10^{-5}, 2^4)$	$(10^{-2}, 2^4)$	$(10^{-1}, 10^0, 2^5)$	$(10^{-5}, 10^{-2}, 2^5)$	$(10^{-1}, 10^{-3}, 2^5)$
	0.0048	0.0067	0.0262	0.0258	0.0096	0.0103	0.009
	32.68	78.31	94.7	94.7	**96.4**	90.05	94.7
Iris	37	16	3	3	2	6	3
(150×4, 3)	-	-	$(10^{-1}, 2^5)$	$(10^{-5}, 2^5)$	$(10^{-1}, 10^{-3}, 2^2)$	$(10^{-1}, 10^{-2}, 2^0)$	$(10^{-1}, 10^{-2}, 2^2)$
	0.0059	0.0051	0.019	0.0197	0.0063	0.007	0.0062
	32.77	76.1	76.58	87.09	87.01	73.79	**87.71**
Seeds	52	33	33	12	12	33	11
(210×7, 3)	-	-	$(10^{-5}, 2^5)$	$(10^{-3}, 2^5)$	$(10^{-5}, 10^{-3}, 2^5)$	$(10^1, 10^3, 2^3)$	$(10^{-1}, 10^2, 2^4)$
	0.0072	0.0127	0.0311	0.0322	0.0123	0.0133	0.0121
	33.98	44	40.76	46.74	46.52	49.05	**53.08**
Teachingeval	36	35	36	37	36	37	36
(151×5, 3)	-	-	$(10^{-3}, 2^2)$	$(10^0, 2^2)$	$(10^4, 10^1, 2^2)$	$(10^1, 10^3, 2^3)$	$(10^4, 10^3, 2^5)$
	0.0047	0.0033	0.021	0.0206	0.0067	0.0074	0.0064
	32.68	54.67	32.68	42.13	47.53	32.68	**54.88**
Tae	37	36	37	35	37	37	37
(150×5, 3)	-	-	$(10^{-5}, 2^{-5})$	$(10^1, 2^2)$	$(10^0, 10^4, 2^3)$	$(10^{-5}, 10^{-5}, 2^{-5})$	$(10^4, 10^2, 2^5)$
	0.0088	0.0021	0.0199	0.0201	0.0068	0.0071	0.0061
	62.75	33.19	63.92	63.92	52.96	54.5	**70.35**
Hayes-roth	27	33	29	29	33	30	24
(132×4, 3)	-	-	$(10^{-5}, 2^{-1})$	$(10^{-5}, 2^{-1})$	$(10^{-3}, 10^{-1}, 2^2)$	$(10^5, 10^{-4}, 2^1)$	$(10^0, 10^2, 2^1)$
	0.0098	0.0026	0.0173	0.017	0.0051	0.0056	0.0049

Table 7.2 (contd.)

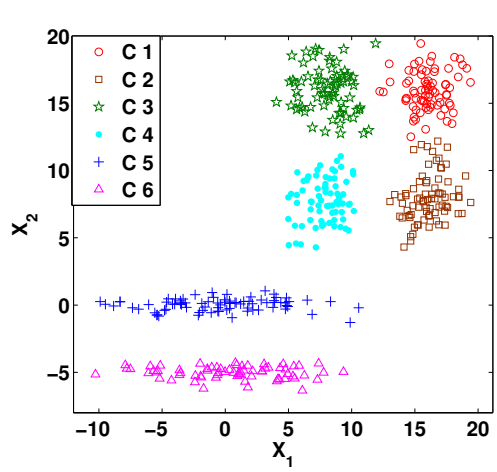
Dataset (Size, clusters)	**FCM** [162] Accuracy # Miss - Time (s)	**kPC** [163] Accuracy # Miss - Time (s)	**TWSVC** [166] Accuracy # Miss (c_1, μ) Time (s)	**TBSVC** [167] Accuracy # Miss $(c_1 = c_2, \mu)$ Time (s)	**LSTWSVC** [168] Accuracy # Miss (c_1, c_2, μ) Time (s)	**FLSTWSVC** [168] Accuracy # Miss (c_1, c_2, μ) Time (s)	**Proposed LSPTSVC** Accuracy # Miss (c_1, c_2, μ) Time (s)
Shuttle (1486×9, 5)	63.25 167 - 0.3162	68.37 145 - 0.565	70.33 150 $(10^{-4}, 2^5)$ 2.5814	70.59 149 $(10^{-5}, 2^4)$ 2.5527	75.73 137 $(10^1, 10^4, 2^5)$ 1.1689	64.66 160 $(10^{-3}, 10^{-5}, 2^2)$ 2.168	**79.96** 132 $(10^{-1}, 10^4, 2^5)$ 1.1518
Libras (360×90, 15)	70.99 89 - 0.4334	71.75 83 - 0.1532	85.68 88 $(10^{-5}, 2^{-1})$ 0.251	81.56 86 $(10^{-3}, 2^0)$ 0.2675	86.41 88 $(10^3, 10^2, 2^{-1})$ 0.0799	**88.6** 87 $(10^{-4}, 10^{-5}, 2^0)$ 0.08	87.15 88 $(10^4, 10^0, 2^4)$ 0.074
Average accuracy	65.23	75.45	76.88	78.87	80.9	80.36	**85.84**
Average rank (Accuracy)	6.1739	4.8696	4.3478	3.7391	3.4565	3.6739	**1.7391**
Average rank (# Miss)	5.8913	4.4348	4.4783	3.6957	3.6522	3.5435	**2.3043**
Win-Tie-Loss	23-0-0	21-1-1	19-2-2	19-2-2	18-1-4	17-2-4	

In terms of mis-clustered data points also, the proposed LSPTSVC performs better than the existing algorithms with an average rank of 2.3043 (Table 7.2). A similar trend is observed for the average ranks of the different algorithms. However, the average rank of FLSTWSVC is lesser than LSTWSVC in terms of mis-clustered data points.

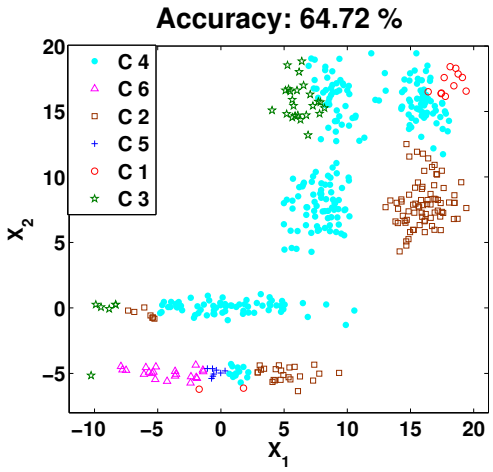
The clusters labelled by all the algorithms using non-linear kernel for Longsquare synthetic dataset are shown in Fig. 7.3. There are 6 clusters in this dataset labelled using non-linear kernel. It is clear from the illustration in Fig. 7.3(f) that LSPTSVC is able to label the clusters better than the other algorithms. Also, FLSTWSVC is showing similar performance in Fig. 7.3(e). A similar trend is visible in Fig. 7.4 for Pathbased dataset, where proposed LSPTSVC outperforms the other algorithms.

7.1.6.3 Statistical analysis

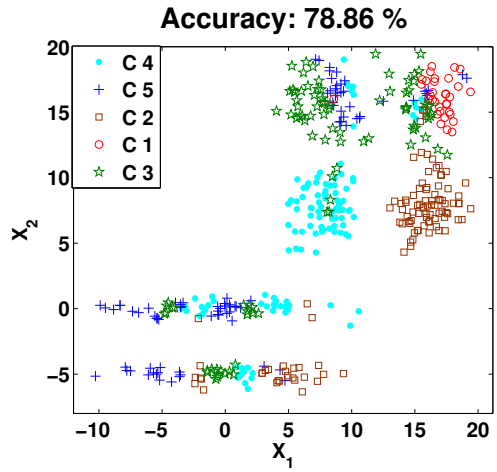
In this section, we check the statistical significance of proposed LSPTSVC with existing techniques in terms of clustering accuracy. We perform the Friedman test [172] with the corresponding Nemenyi post hoc test. Initially, we assume that there is no difference between the methods as the null hypothesis.



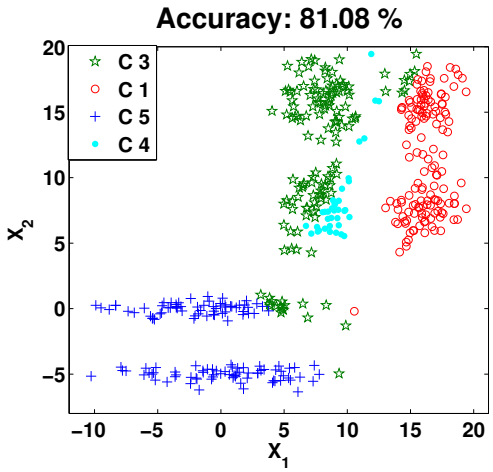
(a) Actual clusters



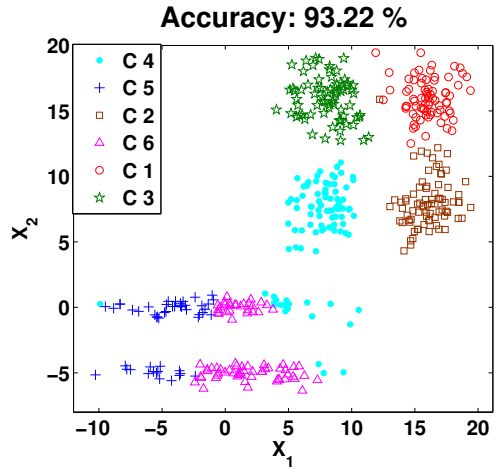
(b) TWSVC



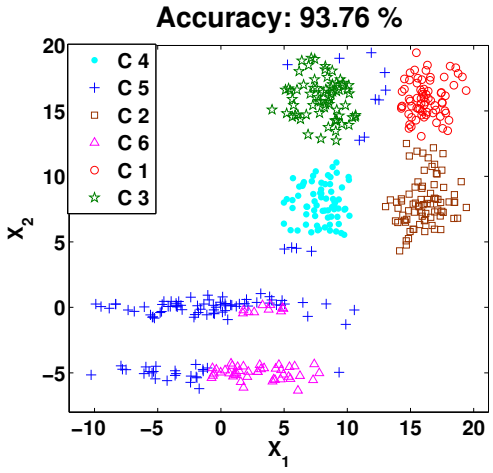
(c) TBSVC



(d) LSTSVC

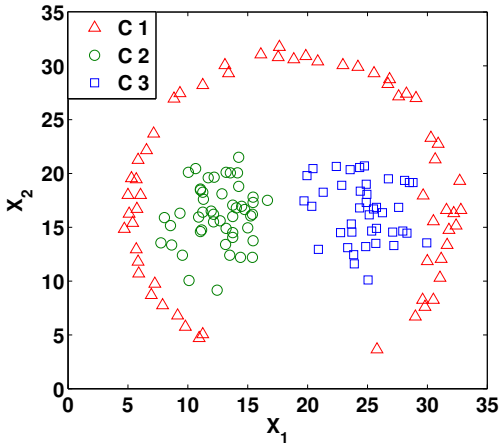


(e) FLSTSVC

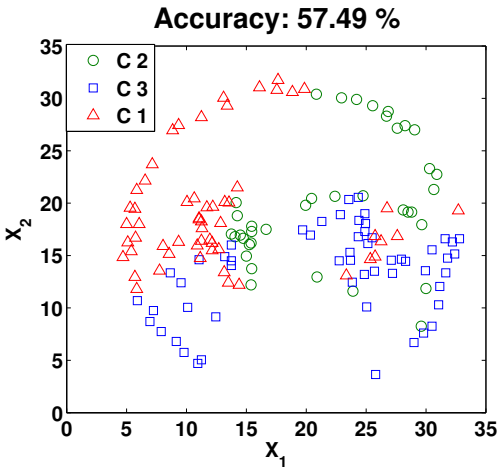


(f) Proposed LSPTSVC

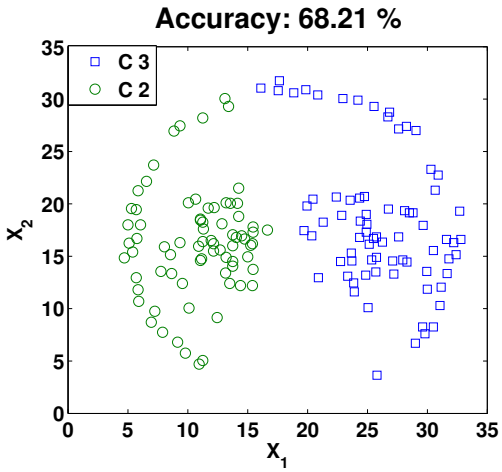
Figure 7.3: Plot showing performance of proposed LSPTSVC in comparison to existing algorithms using RBF kernel for Longsquare dataset. In the legend 'C' means cluster.



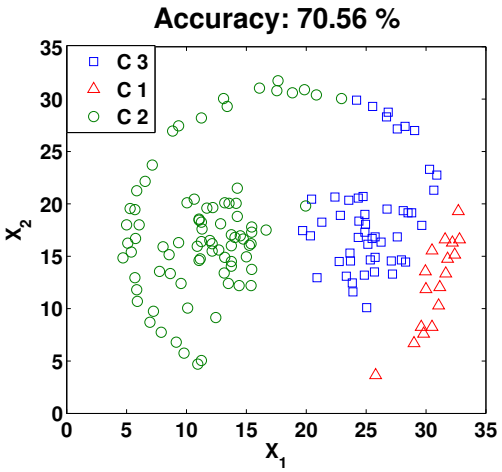
(a) Actual clusters



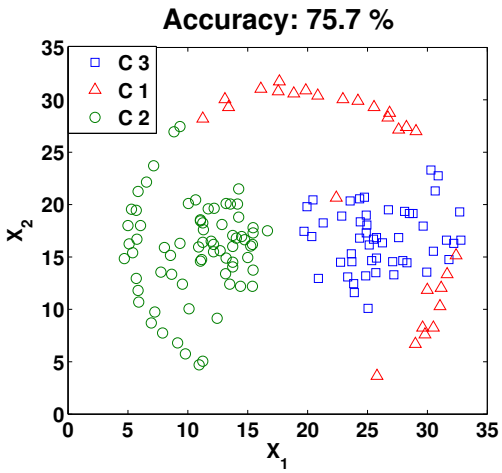
(b) TWSVC



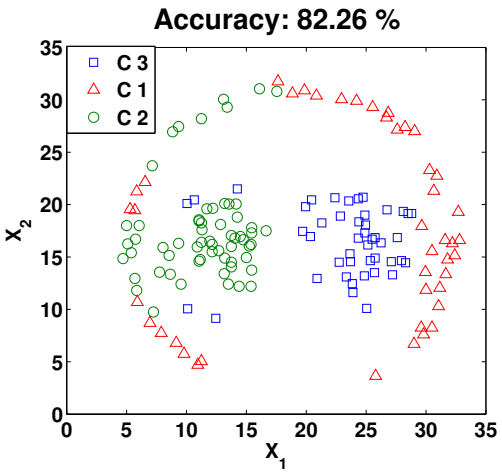
(c) TBSVC



(d) LSTSVC



(e) FLSTSVC



(f) Proposed LSPTSVC

Figure 7.4: Plot showing performance of proposed LSPTSVC in comparison to existing algorithms using RBF kernel for Pathbased dataset. In the legend 'C' means cluster.

I. Linear case:

The χ_F^2 value for Friedman test is calculated using Table 7.1 as 50.7162. The F_F value is calculated as

$$F_F = \frac{(23-1) \times 50.7162}{23 \times (7-1) - 50.7162} = 12.7831.$$

Here, the F-distribution has $\big(7-1, (7-1)(23-1)\big) = (6, 132)$ degrees of freedom. Now, for the level of significance at $\alpha = 0.05$, the critical value of $F(6, 132)$ is 2.1680. Since $F_F = 12.7831 > 2.1680$, we reject the null hypothesis.

Now, to check pairwise difference between the proposed method and existing algorithms, we use the Nemenyi post hoc test. For significant pairwise difference between the methods at $\alpha = 0.10$ level of significance, the average ranks of the methods shown in Table 7.1 should differ by atleast $2.693\sqrt{\frac{7(7+1)}{6\times 23}} = 1.7155$. Table 7.3 shows the pairwise difference between the methods.

Table 7.3: Pairwise significance of the proposed LSPTSVC with existing algorithms.

Linear	**FCM**	**kPC**	**TWSVC**	**TBSVC**	**LSTWSVC**	**FLSTWSVC**
Proposed LSPTSVC	**Yes**	**Yes**	**Yes**	No	No	**Yes**
Non-linear	**FCM**	**kPC**	**TWSVC**	**TBSVC**	**LSTWSVC**	**FLSTWSVC**
Proposed LSPTSVC	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

It can be inferred from Table 7.3 that in the linear case, the proposed LSPTSVC is significantly better than FCM, kPC, TWSVC, and FLSTWSVC algorithms.

II. Non-linear case:

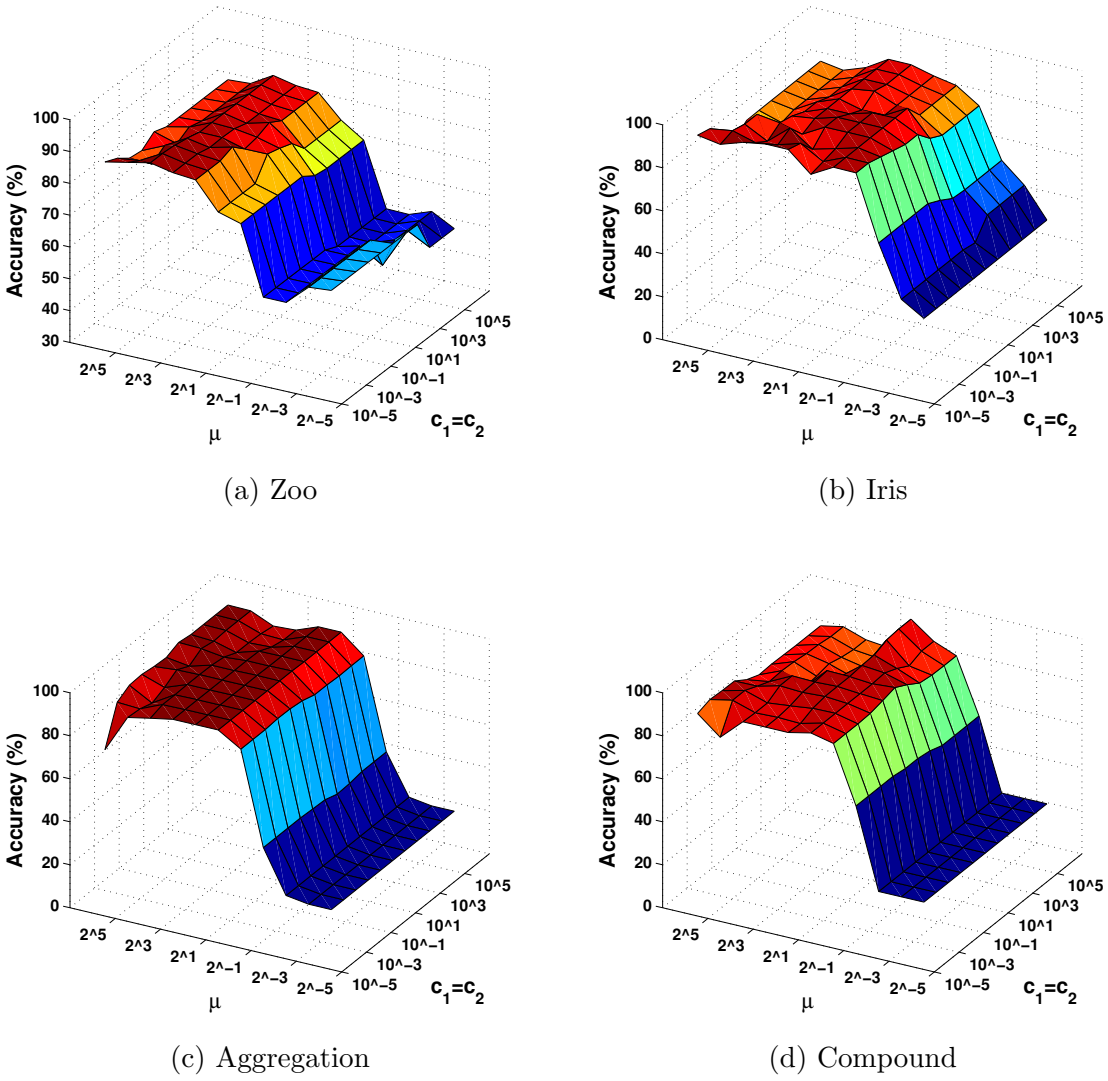
Similar to the linear case, first we calculated the χ^2 value using Table 7.2 as 55.1196.

The F_F value is given as

$$F_F = \frac{(23-1) \times 55.1196}{23 \times (7-1) - 55.1196} = 14.6311.$$

Since $F_F = 14.6311 > 2.1680$, we reject the null hypothesis. Now, similar to linear case, we perform the Nemenyi post hoc test using Table 7.2 to check the pairwise difference between the proposed method and existing algorithms. The results for

the pairwise statistical difference are shown in Table 7.3. It is clear that in terms of clustering accuracy, the proposed LSPTSVC is significantly better than all the existing algorithms for the non-linear case.



(a) Zoo (b) Iris

(c) Aggregation (d) Compound

Figure 7.5: Insensitivity of proposed LSPTSVC for clustering to the user specified parameters $c_1 = c_2$ and μ using RBF kernel for real world benchmark datasets.

To analyze the effect of parameter values on the clustering performance, insensitivity analysis of LSPTSVC is performed for the parameters c and μ. Fig. 7.5 shows the change in accuracy w.r.t. varying values of parameters for real world datasets. It can be observed in Fig. 7.5 that for higher values of μ, the clustering performance is better. This is due to the fact that in non-linear case, μ decides the value of kernel

function, leading to non-linear transformation of data. However, the value of $c_1 = c_2$ does not have any significant effect on the clustering accuracy.

7.1.6.4 Large scale datasets

To show the effectiveness of the proposed LSPTSVC on large sized datasets, experiments are performed on datasets with large number of samples as well as features. A total of 12 large scale datasets are included, where 7 datasets are having large number of samples, while five are having large feature size. The algorithms involving QPPs incur high computation time for large number of samples. Therefore, for large sample datasets, we compared the proposed algorithm with algorithms involving solution of linear equations in Table 7.4. Moreover, linear kernel is used for the comparison.

Table 7.4: Performance comparison on accuracy (%) and training time of proposed LSPTSVC with existing algorithms on large sample datasets using linear kernel. Average rank is based on accuracy.

Dataset (Train size, Test size)	**Clusters**	**LSTWSVC** Accuracy Time (s)	**FLSTWSVC** Accuracy Time (s)	**Proposed LSPTSVC** Accuracy Time (s)
Pendigits (5996×17, 1498×17)	10	**84.76** 1.3836	79.77 1.5956	83.31 1.0081
Penbased (8794×16, 2198×16)	10	75.62 2.6765	71.7 2.9576	**81.59** 2.0746
Letter_10k (8000×16, 2000×16)	26	86.21 6.2636	86.64 6.9988	**91.51** 5.1504
Letter_20k (16000×16, 4000×16)	26	81.16 23.0756	87.54 24.9343	**90.17** 19.9956
Poker_30k (24000×10, 6000×10)	8	**54.71** 13.3279	53.28 16.7914	54.68 10.6532
Poker_40k (32000×10, 8000×10)	9	54.78 30.5703	53.98 36.0595	**55.29** 22.2977
Poker_50k (40000×10, 10000×10)	9	54.41 48.7161	54.51 62.3994	**54.99** 36.6232
Average rank		2.1429	2.5714	**1.2857**

It is evident from Table 7.4 that the proposed LSPTSVC takes least amount of time for large sample datasets. Moreover, the generalization performance of LSPTSVC is also better in most datasets with an average rank of 1.2857. The training time is highest for FLSTWSVC, since it involves the calculation of fuzzy membership matrix.

In case of datasets with large features, we used RBF kernel in all the algorithms. Here, the SMW formula is not used in LSTWSVC, FLSTWSVC, and LSPTSVC, since the feature size is more than the number of samples. Table 7.5 shows the performance for datasets with large feature size. One can observe that the proposed LSPTSVC is efficient on datasets with large number of features. On TTC-3600 dataset, the training time of LSPTSVC is significantly lesser than the other algorithms. Also, the accuracy of LSPTSVC is better for all the datasets. However, the differences in training time of the algorithms are not high. This is due to inclusion of time for generation of kernel matrices in all the algorithms. The time required for generation of kernel matrices is very high in comparison to other steps in the algorithms.

Table 7.5: Performance comparison on accuracy (%) and training time of proposed LSPTSVC with existing algorithms on large feature datasets using RBF kernel. Average rank is based on accuracy.

Dataset (Train size, Test size)	**Clusters**	**TSVC** Accuracy Time (s)	**TBSVC** Accuracy Time (s)	**LSTWSVC** Accuracy Time (s)	**FLSTWSVC** Accuracy Time (s)	**Proposed LSPTSVC** Accuracy Time (s)
Dbworld_emails (52×4702, 12×4702)	2	83.33 0.075	69.7 0.0716	69.7 0.0611	69.7 0.0669	**100** 0.0598
Hydraulic_condition_ps1 (1103×6000, 1102×6000)	2	50.03 61.3294	62.06 61.5112	50.93 60.8006	59.21 60.8788	**62.97** 60.6945
Hydraulic_condition_ps2 (1103×6000, 1102×6000)	2	59.85 60.6696	52.45 61.1182	56.03 60.4428	53.52 60.5068	**70.85** 60.2428
Hydraulic_condition_ps4 (1544×6000, 661×6000)	2	66.39 119.679	66.74 119.708	67.63 118.259	62.33 118.567	**68.91** 117.561
TTC-3600 (2880×7507, 720×7507)	6	68.54 687.917	66.76 660.138	65.09 632.627	60.59 632.861	**69.31** 613.667
Average rank		3	3.4	3.4	4.2	**1**

Figure 7.6: AT&T face recognition data (AT&T Laboratories Cambridge) comprising of 40 individuals.

7.1.7 Applications

In this section, we present some real world applications of the proposed LSPTSVC, along with comparisons to existing algorithms. Experiments are performed on biometric data viz. facial, and facial expression images. Moreover, we also present the application of LSPTSVC on biomedical data. We use ADNI MRI data to evaluate clustering ability of LSPTSVC on Alzheimer's disease. For all the applications, RBF kernel is used in proposed and existing algorithms.

7.1.7.1 Face images

A total of 400 images are included from the AT&T face recognition database shown in Fig. 7.6. The dataset consists of 10 images of 40 individuals, each having dimension of 112×92. 200 images are used in the training as well as testing phase. The dataset is constructed by using all pixel values of an image in one row of the dataset matrix. To avoid overfitting of model, we use principal component analysis (PCA) and class discriminatory ratio (CDR) [9] to reduce the dimension of the dataset to 400×100. The results for face clustering are shown in Fig. 7.7. It is evident that LSPTSVC is able to cluster facial data with better accuracy i.e., 95.53% in comparison to other algorithms.

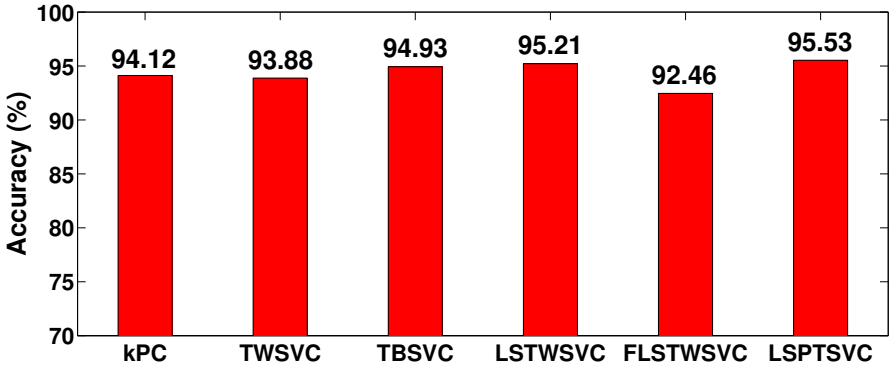


Figure 7.7: Performance comparison of the proposed LSPTSVC with existing algorithms for clustering of AT&T face recognition data.

7.1.7.2 Facial expression images

A total of 210 images are downloaded from the well known JAFFE facial expression database [229] having 30 images of each expression. The dataset contains 7 classes as shown in Fig. 7.8. A total of 140 images are used for training consisting of 20 images of each class, and 70 for testing containing 10 images of each class. The dimension of all the images is 256×256.

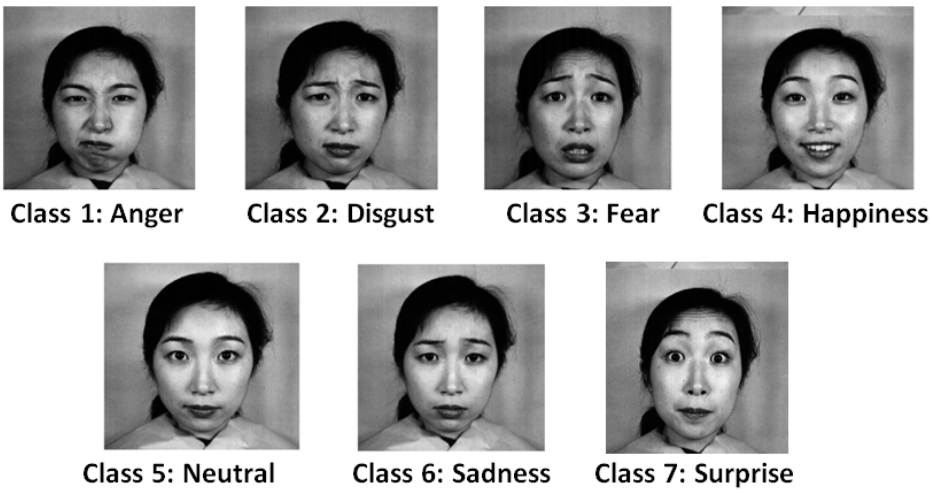


Figure 7.8: Sample images of JAFFE database showing different facial emotions.

In comparison to face recognition, facial expression is a much more difficult problem due to large variations in facial expressions among individuals [231]. However, edge based information is useful for identifying facial expressions [232]. Wavelet transform is

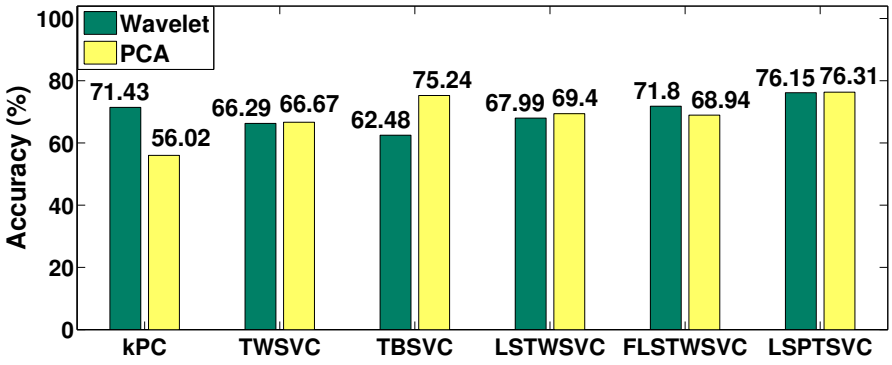


Figure 7.9: Performance comparison of proposed LSPTSVC with existing algorithms for clustering of JAFFE facial expression data.

used for extracting high frequency components [9] responsible for the edges. Therefore, we used PCA along with wavelet transform for dimension reduction, and extraction of useful information for expression detection. Wavelet transform [9] is performed using 'Daeubechies-4' wavelet upto 3 levels of decomposition. Further, CDR is utilized to select the prominent features. After dimension reduction, the size of the dataset becomes 210×50. The results for clustering are shown in Fig. 7.9. In comparison to face recognition, the accuracy is lower in all the algorithms for both features. However, it can be observed that the proposed LSPTSVC is showing highest clustering accuracy for both feature sets i.e., PCA and wavelet for the facial expression dataset.

Other algorithms obtaining high clustering accuracy are TBSVC and FLSTWSVC. This is because TBSVC involves the regularization term, and FLSTWSVC includes fuzzy membership information about the data points.

7.1.7.3 Alzheimer's disease

Alzheimer's disease is an incurable disease affecting 50 million people worldwide [11]. Classification of Alzheimer's disease data is a challenging task [2]. For unlabelled Alzheimer data, clustering is a very useful option. As per our search, there is no work on application of SVM based clustering techniques for Alzheimer's disease data. Therefore, we used Alzheimer's data for clustering by the proposed LSPTSVC and compared with other algorithms.

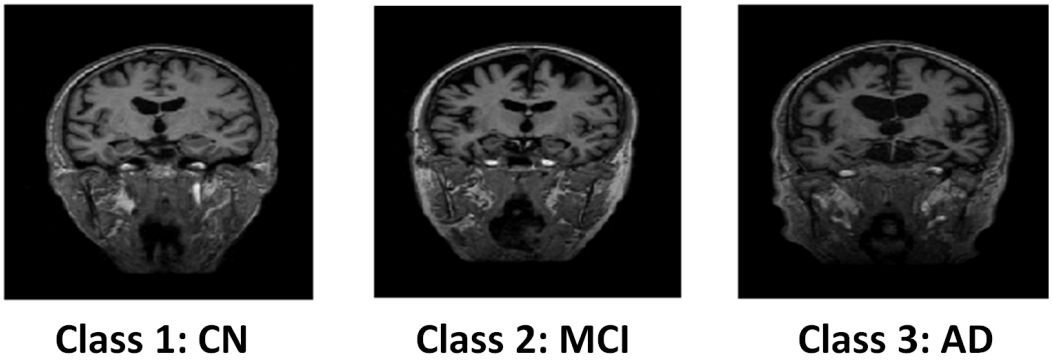


Figure 7.10: MRI images of CN, MCI, and AD subject from ADNI database.

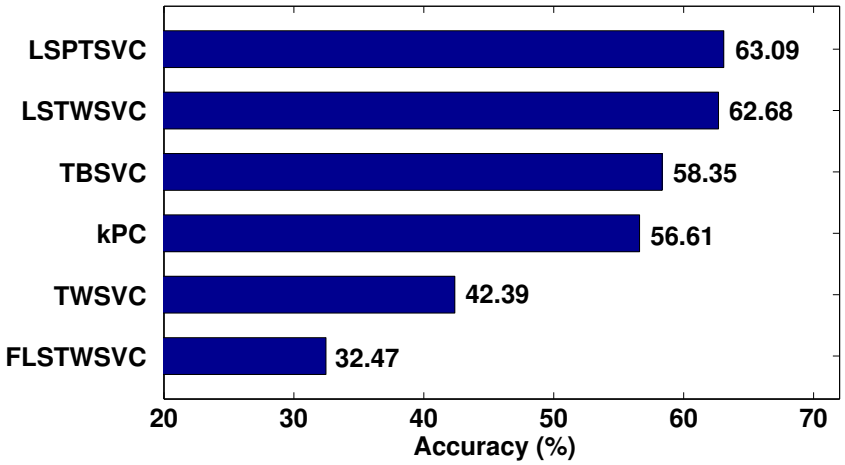


Figure 7.11: Performance comparison of proposed LSPTSVC with existing algorithms for clustering of ADNI Alzheimer's disease data.

The preprocessing and image details are same as in subsection 4.2.2. The images belonging to three categories i.e., control normal (CN), mild cognitive impairment (MCI), and Alzheimer's disease (AD) as shown in Fig. 7.10. 74 images are used for training and 75 for testing. The results are shown in Fig. 7.11.

It is evident that the proposed LSPTSVC is effective in clustering Alzheimer's data in comparison to other algorithms. LSPTSVC obtains a clustering accuracy of 63.09% for CN, MCI and AD subjects. The clustering accuracy of LSPTSVC is similar to previous works [206, 233] on multiclass classification of Alzheimer's data. However, the clustering accuracy of FLSTWSVC is lowest among all the methods. This may be attributed to the presence of outliers in the data. Indeed, the clustering accuracies of all the algorithms are comparatively lower than the applications discussed in the

previous subsections. This is because the data points belonging to the classes MCI and AD are non-linear in their distribution, and are overlapping in nature [28] as shown in Fig. 4.19. Moreover, MCI is an intermediate stage between CN and AD, leading to incorrect labelling of data points [234]. Therefore, the Alzheimer's dataset is difficult to classify [2] or cluster.

7.2 Summary

In this chapter, we proposed a novel twin SVM based unsupervised learning technique. The proposed technique is a projection based clustering algorithm termed as LSPTSVC. The proposed LSPTSVC finds projection axes instead of projection planes for clustering. This is an alternative to the plane based clustering algorithms. The solution of proposed LSPTSVC is obtained by solving a set of linear equations, leading to lesser computational cost. Consequently, no optimization toolbox is required for LSPTSVC. Moreover, LSPTSVC is an efficient algorithm for clustering on datasets with large sample and feature size.

In case of real world applications, LSPTSVC performed better than the existing algorithms. This justifies its applicability for real world applications. In case of Alzheimer's disease, the proposed LSPTSVC has shown significantly better performance, justifying its use for healthcare applications.

The following chapter concludes the works presented in this thesis with possible future directions.

Chapter 8

Conclusions and Future Work

The works presented in this thesis involve novel ideas for the improvement and application of SVM based models. First, the thesis gave a comprehensive review on the various variants of SVM developed in the past. Then, various novel SVM based algorithms are presented primarily for supervised learning. However, one SVM based unsupervised algorithm is proposed for clustering of data. The works presented in this thesis addresses many drawbacks in the existing SVM based algorithms. Also, applications are shown for the proposed algorithms on biomedical datasets. One of the problems addressed in this thesis is class imbalance. This problem is solved by using fuzzy memberships, and with a reduced kernel based approach using universum data. Another problem is of noisy data, which has been dealt by a fuzzy based SVM approach with novel fuzzy functions.

Moreover, reviews are presented for applications of SVM on biomedical data. Various applications of SVM have been discussed, especially for diseases, such as epilepsy, Alzheimer's disease, and breast cancer. The disease which is reviewed extensively in this thesis is Alzheimer's disease. Further, one of the key aspect of this thesis is to develop better universum based SVM algorithms, and apply on biomedical data. The appropriate selection of universum is a research problem. Therefore, we utilized univerum data from the EEG dataset itself, leading to better classification accuracy. We extended USVM algorithm for feature selection by proposing a universum based RFE algorithm, and showed its applicability on Alzheimer's disease data. Since the USVM

based model is transparent and clearly interpretable, it can be used for identifying the features from data. These features can then be analysed for identifying biomarkers in biomedical data of diseases.

Further, to address the drawback of high computation time in USVM based algorithms, we proposed novel variants of USVM to reduce the training time, as well as improve the generalization performance. All the proposed algorithms have been tested on benchmark datasets viz. synthetic, real world, as well as related to biomedical domain. The proposed algorithms showed statistically significant results in these experiments.

In the following subsections, the conclusions of the novel approaches proposed in this thesis is given, followed by a discussion on the possible future directions.

8.1 Conclusions of the proposed works

The conclusions on the different works proposed in this thesis are given under the objectives accomplished in the following:

(i). **Reducing class imbalance:** The problem of class imbalance in SVM learning is addressed in this thesis by proposing two novel algorithms viz. RFLSTSVM-CIL and RUTSVM-CIL. The RFLSTSVM-CIL algorithm is proposed using 2-norm of the slack variable, making the optimization problem strongly convex and implies a globally unique solution. We also proposed a novel fuzzy membership function specifically for class imbalance learning, which gives different range of fuzzy membership values for different imbalanced datasets. The different range of the fuzzy membership function helps in giving proper weights to the data points in different imbalance scenarios. The proposed approach has shown good generalization performance with noisy data as compared to the existing algorithms. From the experiments, it is clear that the proposed RFLSTSVM-CIL approach is having the least ranks for most of the datasets on the basis of AUC, justifying its robustness. Moreover, RFLSTSVM-CIL takes less computation time as compared to the existing fuzzy based algorithms for parallel

and non-parallel support vector machines which justifies its applicability to real world applications.

The second method i.e., proposed RUTSVM-CIL 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. The proposed 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.

(ii). **Review on Alzheimer's disease:** For showing application of SVM on brain disorder, we presented a comprehensive survey on the use of SVM based techniques for Alzheimer's disease. We concluded that SVM has been the most frequently used algorithm for classification of Alzheimer's disease data. As discussed earlier, different variants of SVM have been employed in the past for classification of Alzheimer's. However, it is observed that only 7% of the papers are in the others category. This category involves the algorithms based on SVM which are modified especially for Alzheimer's. Also, it is observed that very few variants of SVM have been applied for AD. It shows the need for research in application of other variants of SVM for AD.

(iii). **Universum learning for brain disorders:** We proposed two novel universum based techniques for brain disorders viz. epilepsy, and Alzheimer's disease. The USVM based technique is proposed for detection of seizure and healthy

EEG signals, while the USVM-RFE technique is proposed with application to feature selection in Alzheimer's disease. In case of seizure detection, on the basis of the experimental results, it can be stated that the proposed universum based approach gives better generalization performance for the classification of EEG signals as compared to the existing approaches. The proposed method of selection of universum points has proved to be a promising approach for the classification of healthy and seizure EEG signals. Also, the effect of outliers on the universum is reduced by using the universum from the EEG dataset itself i.e., the seizure free EEG signal. The distribution of interictal (seizure free) signals provides prior information about the distribution of healthy and seizure signals and also lies in between the two classes. Based on the experimental results, it is evident that universum twin support vector machine (UTSVM) is better in comparison to other support vector machine algorithms for EEG signal classification. Among the different feature extraction techniques, ICA shows the best results using the proposed approach with 99% accuracy.

Moreover, the second proposed method i.e., USVM-RFE provides an improvement over SVM-RFE algorithm by giving prior information about data. On the basis of our analysis on Alzheimer's disease, the proposed USVM-RFE has performed better than SVM-RFE in most of the cases for classification of CN, MCI, and AD subjects. Moreover, we presented an approach of using VBM on training and testing phase separately. This is useful in real world scenarios. We provided an analysis of the feature extraction methods for MRI images i.e., voxel based and volume based features. On the basis of our work on VBM and VolBM features, USVM-RFE achieved relatively better classification accuracy of CN vs AD and MCI vs AD in VolBM as 100% and 73.68% respectively. For CN vs MCI, VBM features provided highest accuracy of 90%. According to our feature analysis, amygdala volume is a prominent discriminative feature for detection of Alzheimer's. One of the important advantages of the proposed universum based algorithm is the global or holistic approach in feature selection as compared to SVM-RFE. Therefore, there is robustness in the proposed

USVM-RFE for each iteration of the feature elimination process.

(iv). **Noise insensitive USVM based classifiers:** To reduce the effect of noise in USVM based algorithms, we used fuzzy based techniques to propose three novel algorithms viz. FUSVM, FUTSVM, and FULSTSVM. First, we proposed a fuzzy based approach for USVM and UTSVM algorithms to remove the effect of outliers. The proposed FUSVM and FUTSVM have shown better generalization performance for most of the datasets. This fuzzy based approach for universum helps in giving prior information to the data in an effective manner. The use of information entropy of the universum points is helpful in giving optimum membership values to the universum data points.

Further, we proposed a novel and efficient fuzzy based learning algorithm, termed as fuzzy universum least squares twin support vector machine (FULSTSVM). The proposed algorithm gives prior information about data distribution to the classifier, and also provides fuzzy membership to the data points and universum. Moreover, the optimization problem of FULSTSVM is solved by a system of linear equations. This makes FULSTSVM efficient in terms of training time. The proposed FULSTSVM is a robust universum based algorithm for classification of data with outliers. Statistical tests on experimental results confirm the significance of the proposed algorithm. Proposed FULSTSVM also performed better on large sized datasets in terms of accuracy, showing its scalability on large datasets. Results on applications i.e., Alzheimer's disease and breast cancer clearly show the applicability of the proposed FULSTSVM for healthcare data.

(v). **Efficient UTSVM based classifiers:** To reduce the computation time of existing universum based techniques, we proposed three novel twin USVM based algorithms viz. AULSTSVM, IUTSVM, and ULSTPMSVM. These techniques also showed improved generalization performance of the model. The proposed AULSTSVM removes the drawback of existing least squares based algorithms w.r.t. computation time. Moreover, the proposed AULSTSVM gives better

generalization performance due to the incorporation of prior information in the training process. Instead of the traditional twin hyperplane based approach for obtaining the decision function, the proposed AULSTSVM needs only one hyperplane for the classification. In contrast to ULSTSVM, AULSTSVM constructs hyperplane for universum data, and utilizes the prior information about data using an angle based approach. Statistical tests confirm the advantage of proposed AULSTSVM in comparison to existing algorithms. The lesser computation time of proposed AULSTSVM is due to the introduction of linear loss in the construction of universum hyperplane. This shows the applicability of proposed AULSTSVM on various real world applications using universum data. Various experiments on large scale datasets clearly show the benefit of the proposed AULSTSVM on large sample datasets. Moreover, an application on Alzheimer's disease data is also presented, where the proposed AULSTSVM obtains highest classification accuracy of 95% for CN vs AD.

A novel and efficient formulation is proposed as universum least squares twin parametric-margin support vector machine (ULSTPMSVM). The proposed algorithms show high generalization performance with lesser training time in comparison to existing algorithms. The formulation of ULSTPMSVM is an alternative approach towards universum based learning. The optimization problem of ULSTPMSVM involves a parametric model, solved by a system of linear equations. In terms of statistical difference in the generalization performance, the proposed ULSTPMSVM turns out to be significantly better than the existing algorithms. Moreover, the proposed ULSTPMSVM performed well for classification of Alzheimer's disease, showing its applicability on real world biomedical applications.

Moreover, to introduce the SRM principle in the formulation of UTSVM, we proposed an improved universum twin support vector machine (IUTSVM). There is no ill-conditioning of the matrices in calculating the inverse in the proposed IUTSVM. The proposed algorithm takes less computation time than

USVM since it solves a pair of QPP. It can be inferred from the experimental results that IUTSVM shows better generalization performance in comparison to the related algorithms.

(vi). **Unsupervised SVM algorithm:** Lastly, we proposed a unsupervised learning algorithm in this thesis using a projection based twin SVM based approach. The proposed LSPTVSC involves a least squares based approach, leading to less computation cost, and finds projection axes instead of projection planes for clustering. This is an alternative to the plane based clustering algorithms. The solution of the proposed LSPTSVC is obtained by solving a set of linear equations, leading to lesser computational cost. Experimental results show that proposed LSPTSVC obtains better clustering accuracy than existing algorithms with lesser training time. Statistical analysis also implies that the proposed algorithm is significantly better than existing algorithms. Moreover, it is showed that LSPTSVC is an efficient algorithm for clustering on datasets with large sample and feature size.

8.2 Future directions

The possible future works are discussed under the different objectives accomplished in this thesis in the following:

(i). **Reducing class imbalance:** The procedure of parameter selection for the fuzzy membership function can be improved by using heuristic based approaches. The fuzzy membership function proposed with RFLSTSVM-CIL can be applied to various applications involving class imbalance. The proposed approach can be extended to multiclass classification, since in most of the multiclass classification problems there is imbalance of data belonging to the different classes.

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 the proposed 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.

(ii). **Review on Alzheimer's disease:** For the classification of AD data, it is observed that researchers have given more importance to the feature extraction phase, and not much to the classification phase. This can be addressed in future research, since novel models can give some new insight in the diagnosis of Alzheimer's. More work is required in formulation of machine learning models which can integrate information from various modalities for early diagnosis of Alzheimer's disease. Moreover, other than the existing models, some novel variants of SVM also need to be developed for Alzheimer's disease as was done in [140]. Also, one can develop and use novel kernel functions for diagnosis of AD using SVM. Such kind of novel models can increase the classification performance of SVM.

(iii). **Universum learning for brain disorders:** The universum based SVM approach needs to be applied to other diseases which are diagnosed using EEG signals with the proper selection of universum. In future, the proposed universum based approach of EEG classification can be improved in terms of computation time. The proposed universum based approach can be extended to multiclass classification of EEG signals using EEG datasets generated with different feature extraction techniques. It is evident from the experimental results that other variants of SVM such as TWSVM and UTSVM give good generalization and computational performance, and thus can be applied for the classification of different EEG signals.

The proposed USVM-RFE can be improved w.r.t. computation time. Also, more research is needed in the proper selection of universum, as it is dependent on the classification problem. Moreover, the proposed USVM-RFE can be modified for multiclass classification problems, and used on various other real world applications.

(iv). **Noise insensitive USVM based classifiers:** The proposed fuzzy based approach for universum support vector machine can be extended to multiclass classification problems, and can be applied in other variants of USVM as well. Moreover, the proposed FUSVM, FUTSVM and FULSTSVM can be improved by implementing new techniques for selecting the universum. The universum data can be selected from a dataset related to a particular application. Moreover, other novel fuzzy membership functions can also be used with the proposed fuzzy based USVM algorithms for various applications.

(v). **Efficient UTSVM based classifiers:** The proposed AULSTSVM, ULSTPMSVM, and IUTSVM algorithms can be applied on real world problems, where universum sample are selected from the dataset itself. More work is needed for the proper generation of universum data in various applications. The proposed algorithms can be applied on other biomedical applications utilizing the benefit of universum learning with less training time. The generalization performance of proposed angle based approach can be improved by the use of multiple kernel learning. Moreover, the proposed AULSTSVM can also be extended for multiclass classification of data. Due to its lesser computation cost, the proposed AULSTSVM can be very effective for multiclass classification.

In future, more work can be done on improving the computation cost of the universum based SVM algorithms. For example, the solution of universum based SVM algorithms can be obtained by using novel iterative methods for solving an unconstrained version of the optimization problem. Moreover, the loss function can be formulated based on different types of smoothing techniques. Also, novel formulations can be proposed in order to reduce the computation

cost of calculating matrix inversions in the solution.

(vi). **Unsupervised SVM algorithm:** In future, the proposed LSPTSVC can be extended for multiple projection axes, and can be applied on other real world clustering problems. In case of Alzheimer's disease, proposed LSPTSVC has shown significantly better performance, justifying its use for other healthcare applications.

The works presented in this thesis provided some significant improvements on existing SVM algorithms. Moreover, the applications on biomedical data puts emphasis on the development and use of SVM based algorithms for such problems. In future more novel SVM based models can be developed to help the diagnosis of various kinds of diseases, which involve very large sized data. This will lead to automated diagnosis of various types of diseases at the early stages, which will lead to a better quality of life for people, especially the elderly population.

Bibliography

[1] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[2] M. Tanveer, B. Richhariya, R.U. Khan, A.H. Rashid, P. Khanna, M. Prasad, and C.T. Lin. Machine learning techniques for the diagnosis of Alzheimer's disease: A review. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(1s):1–35, 2020.

[3] Amin Karami. An anomaly-based intrusion detection system in presence of benign outliers with visualization capabilities. *Expert Systems with Applications*, 108:36–60, 2018.

[4] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg, 1995.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[6] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 2013.

[7] Benoît Magnin, Lilia Mesrob, Serge Kinkingnéhun, Mélanie Pélégrini-Issac, Olivier Colliot, Marie Sarazin, Bruno Dubois, Stéphane Lehéricy, and Habib Benali. Support vector machine-based classification of Alzheimer's disease from whole-brain anatomical MRI. *Neuroradiology*, 51(2):73–83, 2009.

[8] Khin Nandar Win, Kenli Li, Jianguo Chen, Philippe Fournier Viger, and Keqin Li. Fingerprint classification and identification algorithms for criminal investigation: A survey. *Future Generation Computer Systems*, 2019.

[9] Bharat Richhariya and Deepak Gupta. Facial expression recognition using iterative universum twin support vector machine. *Applied Soft Computing*, 76:53–67, 2019.

[10] Jason Weston, Ronan Collobert, Fabian Sinz, Léon Bottou, and Vladimir Vapnik. Inference with the universum. In *Proceedings of the 23rd International Conference on Machine learning*, pages 1009–1016. ACM, 2006.

[11] Christina Patterson. The state of the art of dementia research: New frontiers. *World Alzheimer's Report 2018*, 2018.

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

[13] Daqing Zhang, Jianfeng Xiao, Nannan Zhou, Mingyue Zheng, Xiaomin Luo, Hualiang Jiang, and Kaixian Chen. A genetic algorithm based support vector machine model for blood-brain barrier penetration prediction. *BioMed Research International*, 2015, 2015.

[14] Andrés García-Floriano, Cuauhtémoc López-Martín, Cornelio Yáñez-Márquez, and Alain Abran. Support vector regression for predicting software enhancement effort. *Information and Software Technology*, 97:99–109, 2018.

[15] Nello Cristianini and Bernhard Scholkopf. Support vector machines and kernel methods: The new generation of learning machines. *AI Magazine*, 23(3):31–31, 2002.

[16] Yingjie Tian and Zhiquan Qi. Review on: Twin support vector machines. *Annals of Data Science*, 1(2):253–277, 2014.

[17] Johan A.K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

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

[19] Olivier Chapelle, Alekh Agarwal, Fabian H Sinz, and Bernhard Schölkopf. An analysis of inference with the universum. In *Advances in Neural Information Processing Systems*, pages 1369–1376, 2008.

[20] B. Richhariya and M. Tanveer. EEG signal classification using universum support vector machine. *Expert Systems with Applications*, 106:169–182, 2018.

[21] Yitian Xu, Mei Chen, Zhiji Yang, and Guohui Li. ν-twin support vector machine with universum data for classification. *Applied Intelligence*, 44(4):956–968, 2016.

[22] Wen Long, Ye-ran Tang, and Ying-jie Tian. Investor sentiment identification based on the universum SVM. *Neural Computing and Applications*, 30(2):661–670, 2018.

[23] Abdulhamit Subasi and M. Ismail Gursoy. EEG signal classification using PCA, ICA, LDA and support vector machines. *Expert Systems with Applications*, 37(12):8659–8666, 2010.

[24] Prashanthi Vemuri and Clifford R. Jack. Role of structural MRI in Alzheimer's disease. *Alzheimer's Research & Therapy*, 2(4):23, 2010.

[25] B. Richhariya and M. Tanveer. A robust fuzzy least squares twin support vector machine for class imbalance learning. *Applied Soft Computing*, 71:418–432, 2018.

[26] B. Richhariya and M. Tanveer. A reduced universum twin support vector machine for class imbalance learning. *Pattern Recognition*, 102:107150, 2020.

[27] Zhiquan Qi, Yingjie Tian, and Yong Shi. Twin support vector machine with universum data. *Neural Networks*, 36:112–119, 2012.

[28] B. Richhariya, M. Tanveer, A.H. Rashid, and Alzheimer's Disease Neuroimaging Initiative. Diagnosis of Alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE). *Biomedical Signal Processing and Control*, 59:101903, 2020.

[29] B. Richhariya and M. Tanveer. A fuzzy universum support vector machine based on information entropy. In M. Tanveer and Ram Bilas Pachori, editors, *Machine Intelligence and Signal Analysis*, volume 748, pages 569–582. Springer Singapore, 2019.

[30] B. Richhariya, M. Tanveer, and Alzheimer's Disease Neuroimaging Initiative. A fuzzy universum least squares twin support vector machine (FULSTSVM). *Neural Computing and Applications*, https://doi.org/10.1007/s00521-021-05721-4, 2021.

[31] B. Richhariya, M. Tanveer, and Alzheimer's Disease Neuroimaging Initiative. An efficient angle based universum least squares twin support vector machine for pattern classification. *ACM Transactions on Internet Technology (TOIT)*, https://doi.org/10.1145/3387131, 2020.

[32] B. Richhariya and M. Tanveer. Universum least squares twin parametric-margin support vector machine. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[33] B. Richhariya, A. Sharma, and M. Tanveer. Improved universum twin support vector machine. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2045–2052. IEEE, 2018.

[34] B. Richhariya, M. Tanveer, and Alzheimer's Disease Neuroimaging Initiative. Least squares projection twin support vector clustering (LSPTSVC). *Information Sciences*, 533:1–23, 2020.

[35] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[36] Naiyang Deng, Yingjie Tian, and Chunhua Zhang. *Support vector machines: optimization based theory, algorithms, and extensions*. Chapman and Hall/CRC, 2012.

[37] Divya Tomar and Sonali 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, 2015.

[38] Xiaochen Zhang, Dongxiang Jiang, Te Han, Nanfei Wang, Wenguang Yang, and Yizhou Yang. Rotating machinery fault diagnosis for imbalanced data based on fast clustering algorithm and support vector machine. *Journal of Sensors*, 2017, 2017.

[39] Divya Tomar and Sonali Agarwal. Prediction of defective software modules using class imbalance learning. *Applied Computational Intelligence and Soft Computing*, 2016, 2016.

[40] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, volume 3. JHU press, 2012.

[41] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 13(2):464–471, 2002.

[42] Rukshan Batuwita and Vasile Palade. FSVM-CIL: Fuzzy support vector machines for class imbalance learning. *IEEE Transactions on Fuzzy Systems*, 18(3):558–571, 2010.

[43] Gilles Cohen, Melanie Hilario, and Christian Pellegrini. One-class support vector machines with a conformal kernel. A case study in handling class imbalance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern*

Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pages 850–858. Springer, 2004.

[44] Benjamin X. Wang and Nathalie Japkowicz. Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, 25(1):1–20, 2010.

[45] Xiaoqing Gu, Tongguang Ni, and Hongyuan Wang. New fuzzy support vector machine for the class imbalance problem in medical datasets classification. *The Scientific World Journal*, 2014, 2014.

[46] Yong Zhang, Panpan Fu, Wenzhe Liu, and Guolong Chen. Imbalanced data classification based on scaling kernel-based support vector machine. *Neural Computing and Applications*, 25(3-4):927–935, 2014.

[47] Qiang Wang. A hybrid sampling SVM approach to imbalanced data classification. In *Abstract and Applied Analysis*, volume 2014. Hindawi, 2014.

[48] Xiaopeng Hua and Shifei Ding. Weighted least squares projection twin support vector machines with local information. *Neurocomputing*, 160:228–237, 2015.

[49] Javad Salimi Sartakhti, Homayun Afrabandpey, and Nasser Ghadiri. Fuzzy least squares twin support vector machines. *arXiv preprint arXiv:1505.05451*, 2015.

[50] Talayeh Razzaghi, Oleg Roderick, Ilya Safro, and Nicholas Marko. Multilevel weighted support vector machine for classification on healthcare data with missing values. *PloS One*, 11(5):e0155119, 2016.

[51] Hong-Liang Dai. Class imbalance learning via a fuzzy total margin based support vector machine. *Applied Soft Computing*, 31:172–184, 2015.

[52] SungHwan Kim. Weighted k-means support vector machine for cancer prediction. *Springerplus*, 5(1):1–11, 2016.

[53] Divya Tomar, Shubham Singhal, and Sonali Agarwal. Weighted least square twin support vector machine for imbalanced dataset. *International Journal of Database Theory and Application*, 7(2):25–36, 2014.

[54] Divya Tomar and Sonali Agarwal. An effective weighted multi-class least squares twin support vector machine for imbalanced data classification. *International Journal of Computational Intelligence Systems*, 8(4):761–778, 2015.

[55] Qi Fan, Zhe Wang, Dongdong Li, Daqi Gao, and Hongyuan Zha. Entropy-based fuzzy support vector machine for imbalanced datasets. *Knowledge-Based Systems*, 115:87–99, 2017.

[56] Tingting Gao, Yingjie Tian, Xiaojian Shao, and Naiyang Deng. Accurate prediction of translation initiation sites by universum SVM. In *The 2nd International Symposium on Optimization and Systems Biology*, pages 279–286. Citeseer, 2008.

[57] Xiaoke Hao and Daoqiang Zhang. Ensemble universum SVM learning for multimodal classification of Alzheimer's disease. In *International Workshop on Machine Learning in Medical Imaging*, pages 227–234. Springer, 2013.

[58] Hojjat Adeli, Ziqin Zhou, and Nahid Dadmehr. Analysis of EEG records in an epileptic patient using wavelet transform. *Journal of Neuroscience Methods*, 123(1):69–87, 2003.

[59] O.A. Rosso, Wendy Hyslop, R. Gerlach, R.L.L. Smith, J.A.P. Rostas, and Mick Hunter. Quantitative EEG analysis of the maturational changes associated with childhood absence epilepsy. *Physica A: Statistical Mechanics and its Applications*, 356(1):184–189, 2005.

[60] Hasan Ocak. Optimal classification of epileptic seizures in EEG using wavelet analysis and genetic algorithm. *Signal Processing*, 88(7):1858–1867, 2008.

[61] Inan Güler and Elif Derya Übeyli. Adaptive neuro-fuzzy inference system for classification of EEG signals using wavelet coefficients. *Journal of Neuroscience Methods*, 148(2):113–121, 2005.

[62] Yuliang Ma, Xiaohui Ding, Qingshan She, Zhizeng Luo, Thomas Potter, and Yingchun Zhang. Classification of motor imagery EEG signals with support

vector machines and particle swarm optimization. *Computational and Mathematical Methods in Medicine*, 2016, 2016.

[63] Yinxia Liu, Weidong Zhou, Qi Yuan, and Shuangshuang Chen. Automatic seizure detection using wavelet transform and SVM in long-term intracranial EEG. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 20(6):749–755, 2012.

[64] Nicoletta Nicolaou and Julius Georgiou. Detection of epileptic electroencephalogram based on permutation entropy and support vector machines. *Expert Systems with Applications*, 39(1):202–209, 2012.

[65] M. Zavar, S. Rahati, M-R. Akbarzadeh-T, and H. Ghasemifard. Evolutionary model selection in a wavelet-based support vector machine for automated seizure detection. *Expert Systems with Applications*, 38(9):10751–10758, 2011.

[66] Lei Guo, Youxi Wu, Lei Zhao, Ting Cao, Weili Yan, and Xueqin Shen. Classification of mental task from EEG signals using immune feature weighted support vector machines. *IEEE Transactions on Magnetics*, 47(5):866–869, 2010.

[67] Yan Li and Peng Wen. Classification of EEG signals using sampling techniques and least square support vector machines. In *International Conference on Rough Sets and Knowledge Technology*, pages 375–382. Springer, 2009.

[68] Varun Bajaj and Ram Bilas Pachori. Classification of seizure and nonseizure EEG signals using empirical mode decomposition. *IEEE Transactions on Information Technology in Biomedicine*, 16(6):1135–1142, 2011.

[69] Siuly, Yan Li, and Peng (Paul) Wen. Clustering technique-based least square support vector machine for EEG signal classification. *Computer Methods and Programs in Biomedicine*, 104(3):358–372, 2011.

[70] Inan Guler and Elif Derya Ubeyli. Multiclass support vector machines for EEG-signals classification. *IEEE Transactions on Information Technology in Biomedicine*, 11(2):117–126, 2007.

[71] Yitian Xu, Mei Chen, and Guohui Li. Least squares twin support vector machine with universum data for classification. *International Journal of Systems Science*, 47(15):3637–3645, 2016.

[72] Rudimar L. Frozza, Mychael V. Lourenco, and Fernanda G. De Felice. Challenges for Alzheimer's disease therapy: Insights from novel mechanisms beyond memory defects. *Frontiers in Neuroscience*, 12:37, 2018.

[73] Randall J. Bateman, Paul S. Aisen, Bart De Strooper, Nick C. Fox, Cynthia A. Lemere, John M. Ringman, Stephen Salloway, Reisa A. Sperling, Manfred Windisch, and Chengjie Xiong. Autosomal-dominant Alzheimer's disease: A review and proposal for the prevention of Alzheimer's disease. *Alzheimer's Research & Therapy*, 3(1):1, 2011.

[74] Tingyan Wang, Robin G. Qiu, and Ming Yu. Predictive modeling of the progression of Alzheimer's disease with recurrent neural networks. *Scientific Reports*, 8, 2018.

[75] Christos Davatzikos, Susan M. Resnick, X. Wu, P. Parmpi, and Christopher M. Clark. Individual patient diagnosis of AD and FTD via high-dimensional pattern classification of MRI. *NeuroImage*, 41(4):1220–1227, 2008.

[76] Saima Rathore, Mohamad Habes, Muhammad Aksam Iftikhar, Amanda Shacklett, and Christos Davatzikos. A review on neuroimaging-based classification studies and associated feature extraction methods for Alzheimer's disease and its prodromal stages. *NeuroImage*, 155:530–548, 2017.

[77] James D. Doecke, Simon M. Laws, Noel G. Faux, et al. Blood-based protein biomarkers for diagnosis of Alzheimer disease. *Archives of Neurology*, 69(10):1318–1325, 2012.

[78] Maria Paraskevaidi, Camilo L.M. Morais, Diane E. Halliwell, David M.A. Mann, David Allsop, Pierre L. Martin-Hirsch, and Francis L. Martin. Raman spec-

troscopy to diagnose Alzheimer's disease and dementia with lewy bodies in blood. *ACS Chemical Neuroscience*, 9(11):2786–2794, 2018.

[79] Clifford R. Jack Jr., Josephine Barnes, Matt A. Bernstein, et al. Magnetic resonance imaging in Alzheimer's Disease Neuroimaging Initiative 2. *Alzheimer's & Dementia*, 11(7):740–756, 2015.

[80] Michael W. Weiner, Dallas P. Veitch, Paul S. Aisen, et al. Recent publications from the Alzheimer's Disease Neuroimaging Initiative: Reviewing progress toward improved ad clinical trials. *Alzheimer's & Dementia*, 13(4):e1–e85, 2017.

[81] Dallas P. Veitch, Michael W. Weiner, Paul S. Aisen, et al. Understanding disease progression and improving Alzheimer's disease clinical trials: Recent highlights from the Alzheimer's Disease Neuroimaging Initiative. *Alzheimer's & Dementia*, 2018.

[82] Motonobu Fujishima, Atsushi Kawaguchi, Norihide Maikusa, Ryozo Kuwano, Takeshi Iwatsubo, and Hiroshi Matsuda. Sample size estimation for Alzheimer's disease trials from Japanese ADNI serial magnetic resonance imaging. *Journal of Alzheimer's Disease*, 56(1):75–88, 2017.

[83] Takeshi Iwatsubo. Japanese Alzheimer's Disease Neuroimaging Initiative: Present status and future. *Alzheimer's & Dementia*, 6(3):297–299, 2010.

[84] K. Kazemi and N. Noorizadeh. Quantitative comparison of SPM, FSL, and brainsuite for brain MR image segmentation. *Journal of Biomedical Physics & Engineering*, 4(1):13, 2014.

[85] Anders M. Dale, Bruce Fischl, and Martin I. Sereno. Cortical surface-based analysis: I. segmentation and surface reconstruction. *NeuroImage*, 9(2):179–194, 1999.

[86] Saruar Alam, Goo-Rak Kwon, and Alzheimer's Disease Neuroimaging Initiative. Alzheimer disease classification using KPCA, LDA, and multi-kernel learning

SVM. *International Journal of Imaging Systems and Technology*, 27(2):133–143, 2017.

[87] Sabina Tangaro, Annarita Fanizzi, Nicola Amoroso, Roberto Bellotti, and Alzheimer's Disease Neuroimaging Initiative. A fuzzy-based system reveals Alzheimer's disease onset in subjects with mild cognitive impairment. *Physica Medica*, 38:36–44, 2017.

[88] M. Termenon, Manuel Grana, A. Besga, J. Echeveste, and A. Gonzalez-Pinto. Lattice independent component analysis feature selection on diffusion weighted imaging for Alzheimer's disease classification. *Neurocomputing*, 114:132–141, 2013.

[89] Elaheh Moradi, Antonietta Pepe, Christian Gaser, Heikki Huttunen, Jussi Tohka, and Alzheimer's Disease Neuroimaging Initiative. Machine learning framework for early MRI-based Alzheimer's conversion prediction in MCI subjects. *NeuroImage*, 104:398–412, 2015.

[90] Enrico Pellegrini, Lucia Ballerini, Maria del C. Valdes Hernandez, et al. Machine learning of neuroimaging for assisted diagnosis of cognitive impairment and dementia: A systematic review. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, 10:519–535, 2018.

[91] Halil Bisgin, Tanmay Bera, Hongjian Ding, et al. Comparing SVM and ANN based machine learning methods for species identification of food contaminating beetles. *Scientific Reports*, 8, 2018.

[92] Zhenghao Shi, Lifeng He, Kenji Suzuki, Tsuyoshi Nakamura, and Hidenori Itoh. Survey on neural networks used for medical image processing. *International Journal of Computational Science*, 3(1):86, 2009.

[93] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, 19:221–248, 2017.

[94] Heung-Il Suk, Seong-Whan Lee, Dinggang Shen, and Alzheimer's Disease Neuroimaging Initiative. Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis. *NeuroImage*, 101:569–582, 2014.

[95] Manhua Liu, Daoqiang Zhang, Dinggang Shen, and Alzheimer's Disease Neuroimaging Initiative. Ensemble sparse classification of Alzheimer's disease. *NeuroImage*, 60(2):1106–1116, 2012.

[96] David Glenn Clark, Paula M. McLaughlin, Ellen Woo, et al. Novel verbal fluency scores and structural brain imaging for prediction of cognitive outcome in mild cognitive impairment. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, 2:113–122, 2016.

[97] Claudia Plant, Stefan J. Teipel, Annahita Oswald, Christian Böhm, Thomas Meindl, Janaina Mourao-Miranda, Arun W Bokde, Harald Hampel, and Michael Ewers. Automated detection of brain atrophy patterns based on MRI for the prediction of Alzheimer's disease. *NeuroImage*, 50(1):162–174, 2010.

[98] Iman Beheshti, Hasan Demirel, Hiroshi Matsuda, and Alzheimer's Disease Neuroimaging Initiative. Classification of Alzheimer's disease and prediction of mild cognitive impairment-to-Alzheimer's conversion from structural magnetic resource imaging using feature ranking and a genetic algorithm. *Computers in Biology and Medicine*, 83:109–119, 2017.

[99] Rémi Cuingnet, Emilie Gerardin, Jérôme Tessieras, Guillaume Auzias, Stéphane Lehéricy, Marie-Odile Habert, Marie Chupin, Habib Benali, Olivier Colliot, and Alzheimer's Disease Neuroimaging Initiative. Automatic classification of patients with Alzheimer's disease from structural MRI: A comparison of ten methods using the ADNI database. *NeuroImage*, 56(2):766–781, 2011.

[100] Laurence O'Dwyer, Franck Lamberton, Arun L. W. Bokde, Michael Ewers, Yetunde O. Faluyi, Colby Tanner, Bernard Mazoyer, Desmond O'Neill, Máiréad Bartley, D. Rónán Collins, Tara Coughlan, David Prvulovic, and Harald Ham-

pel. Using support vector machines with multiple indices of diffusion for automated classification of mild cognitive impairment. *PloS One*, 7(2):e32441, 2012.

[101] Esteve Gallego-Jutglà, Jordi Solé-Casals, François-Benoît Vialatte, Mohamed Elgendi, Andrzej Cichocki, and Justin Dauwels. A hybrid feature selection approach for the early diagnosis of Alzheimer's disease. *Journal of Neural Engineering*, 12(1):016018, 2015.

[102] Yingying Zhu, Xiaofeng Zhu, Minjeong Kim, Dinggang Shen, and Guorong Wu. Early diagnosis of Alzheimer's disease by joint feature selection and classification on temporally structured support vector machine. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 264–272. Springer, 2016.

[103] Sandeep Chaplot, L.M. Patnaik, and N.R. Jagannathan. Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network. *Biomedical Signal Processing and Control*, 1(1):86–92, 2006.

[104] Salim Lahmiri and Mounir Boukadoum. New approach for automatic classification of Alzheimer's disease, mild cognitive impairment and healthy brain magnetic resonance images. *Healthcare Technology Letters*, 1(1):32–36, 2014.

[105] Rupali S. Kamathe and Kalyani R. Joshi. A novel method based on independent component analysis for brain MR image tissue classification into CSF, WM and GM for atrophy detection in Alzheimer's disease. *Biomedical Signal Processing and Control*, 40:41–48, 2018.

[106] Yong Fan, Susan M. Resnick, Xiaoying Wu, and Christos Davatzikos. Structural and functional biomarkers of prodromal Alzheimer's disease: A high-dimensional pattern classification study. *NeuroImage*, 41(2):277–285, 2008.

[107] Juergen Dukart, Karsten Mueller, Henryk Barthel, Arno Villringer, Osama Sabri, Matthias Leopold Schroeter, and Alzheimer's Disease Neuroimaging Initiative. Meta-analysis based SVM classification enables accurate detection of

Alzheimer's disease across different clinical centers using FDG-PET and MRI. *Psychiatry Research: Neuroimaging*, 212(3):230–236, 2013.

[108] Ignacio Álvarez, Míriam López, Juan Manuel Górriz, Javier Ramírez, Diego Salas-Gonzalez, Carlos García Puntonet, and Fermín Segovia. Automatic classification system for the diagnosis of Alzheimer disease using component-based SVM aggregations. In *International Conference on Neural Information Processing*, pages 402–409. Springer, 2008.

[109] F. Segovia, J.M. Górriz, J. Ramírez, D. Salas-González, I. Álvarez, M. López, R. Chaves, and P. Padilla. Classification of functional brain images using a gmm-based multi-variate approach. *Neuroscience Letters*, 474(1):58–62, 2010.

[110] Seyed Hani Hojjati, Ata Ebrahimzadeh, Ali Khazaee, Abbas Babajani-Feremi, and Alzheimer's Disease Neuroimaging Initiative. Predicting conversion from mci to ad using resting-state fMRI, graph theoretical approach and SVM. *Journal of Neuroscience Methods*, 282:69–80, 2017.

[111] Jinhua Sheng, Bocheng Wang, Qiao Zhang, Qingqiang Liu, Yangjie Ma, Weixiang Liu, Meiling Shao, and Bin Chen. A novel joint HCPMMP method for automatically classifying Alzheimer's and different stage mci patients. *Behavioural Brain Research*, 2019.

[112] Wook Lee, Byungkyu Park, and Kyungsook Han. Classification of diffusion tensor images for the early detection of Alzheimer's disease. *Computers in Biology and Medicine*, 43(10):1313–1320, 2013.

[113] Ying-Teng Zhang and Shen-Quan Liu. Individual identification using multimetric of DTI in Alzheimer's disease and mild cognitive impairment. *Chinese Physics B*, 27(8):088702, 2018.

[114] Sven Haller, Pascal Missonnier, F.R. Herrmann, Cristelle Rodriguez, M-P Deiber, Duy Nguyen, Gabriel Gold, K-O Lovblad, and Panteleimon Giannakopoulos. Individual classification of mild cognitive impairment subtypes by

support vector machine analysis of white matter DTI. *American Journal of Neuroradiology*, 34(2):283–291, 2013.

[115] Andrés Ortiz, Juan M. Górriz, Javier Ramírez, Francisco Jesús Martínez-Murcia, and Alzheimer's Disease Neuroimaging Initiative. LVQ-SVM based CAD tool applied to structural MRI for the diagnosis of the Alzheimer's disease. *Pattern Recognition Letters*, 34(14):1725–1733, 2013.

[116] Yudong Zhang, Shuihua Wang, Preetha Phillips, Zhengchao Dong, Genlin Ji, and Jiquan Yang. Detection of Alzheimer's disease and mild cognitive impairment based on structural volumetric MR images using 3D-DWT and WTA-KSVM trained by PSOTVAC. *Biomedical Signal Processing and Control*, 21:58–73, 2015.

[117] Prashanthi Vemuri, Jeffrey L. Gunter, Matthew L. Senjem, Jennifer L. Whitwell, Kejal Kantarci, David S. Knopman, Bradley F. Boeve, Ronald C. Petersen, and Clifford R. Jack Jr. Alzheimer's disease diagnosis in individual subjects using structural mr images: validation studies. *NeuroImage*, 39(3):1186–1197, 2008.

[118] Lilia Mesrob, Benoit Magnin, Olivier Colliot, Marie Sarazin, Valérie Hahn-Barma, Bruno Dubois, Patrick Gallinari, Stéphane Lehéricy, Serge Kinkingnéhun, and Habib Benali. Identification of atrophy patterns in Alzheimer's disease based on SVM feature selection and anatomical parcellation. In *International Workshop on Medical Imaging and Virtual Reality*, pages 124–132. Springer, 2008.

[119] Emilie Gerardin, Gaël Chételat, Marie Chupin, et al. Multidimensional classification of hippocampal shape features discriminates Alzheimer's disease and mild cognitive impairment from normal aging. *NeuroImage*, 47(4):1476–1486, 2009.

[120] R. Chaves, J. Ramírez, J.M. Górriz, M. López, D. Salas-Gonzalez, I. Alvarez, and F. Segovia. SVM-based computer-aided diagnosis of the Alzheimer's disease

using t-test nmse feature selection with feature correlation weighting. *Neuroscience Letters*, 461(3):293–297, 2009.

[121] Javier Ramírez, Juan Manuel Górriz, Míriam López, Diego Salas-Gonzalez, Ignacio Álvarez, Fermín Segovia, and Carlos García Puntonet. Early detection of the Alzheimer disease combining feature selection and kernel machines. In *International Conference on Neural Information Processing*, pages 410–417. Springer, 2008.

[122] Andrés Ortiz, Jorge Munilla, Ignacio Álvarez-Illán, Juan M. Górriz, Javier Ramírez, and Alzheimer's Disease Neuroimaging Initiative. Exploratory graphical models of functional and structural connectivity patterns for Alzheimer's disease diagnosis. *Frontiers in Computational Neuroscience*, 9:132, 2015.

[123] P Padilla, J.M. Górriz, J. Ramírez, E.W. Lang, R. Chaves, F. Segovia, M. López, D. Salas-González, and I. Álvarez. Analysis of SPECT brain images for the diagnosis of Alzheimer's disease based on NMF for feature extraction. *Neuroscience Letters*, 479(3):192–196, 2010.

[124] Ahmed Abdulkadir, Bénédicte Mortamet, Prashanthi Vemuri, Clifford R. Jack Jr, Gunnar Krueger, Stefan Klöppel, and Alzheimer's Disease Neuroimaging Initiative. Effects of hardware heterogeneity on the performance of SVM Alzheimer's disease classifier. *NeuroImage*, 58(3):785–792, 2011.

[125] Daniel Schmitter, Alexis Roche, Bénédicte Maréchal, et al. An evaluation of volume-based morphometry for prediction of mild cognitive impairment and Alzheimer's disease. *NeuroImage: Clinical*, 7:7–17, 2015.

[126] Maciej Plocharski, Lasse Riis Østergaard, and Alzheimer's Disease Neuroimaging Initiative. Extraction of sulcal medial surface and classification of Alzheimer's disease using sulcal features. *Computer Methods and Programs in Biomedicine*, 133:35–44, 2016.

[127] Fermín Segovia, J.M. Górriz, Javier Ramírez, Diego Salas-Gonzalez, and Ignacio Álvarez. Early diagnosis of Alzheimer's disease based on partial least squares and support vector machine. *Expert Systems with Applications*, 40(2):677–683, 2013.

[128] Alessandra Retico, Paolo Bosco, Piergiorgio Cerello, Elisa Fiorina, Andrea Chincarini, and Maria Evelina Fantacci. Predictive models based on support vector machines: Whole-brain versus regional analysis of structural MRI in the Alzheimer's disease. *Journal of Neuroimaging*, 25(4):552–563, 2015.

[129] Antonio R. Hidalgo-Muñoz, Javier Ramírez, Juan M. Górriz, and Pablo Padilla. Regions of interest computed by SVM wrapped method for Alzheimer's disease examination from segmented MRI. *Frontiers in Aging Neuroscience*, 6:20, 2014.

[130] Laila Khedher, Javier Ramírez, Juan Manuel Górriz, Abdelbasset Brahim, and I.A. Illán. Independent component analysis-based classification of Alzheimer's disease from segmented MRI data. In *International Work-Conference on the Interplay between Natural and Artificial Computation*, pages 78–87. Springer, 2015.

[131] Laila Khedher, Ignacio A. Illán, Juan M. Górriz, Javier Ramírez, Abdelbasset Brahim, and Anke Meyer-Baese. Independent component analysis-support vector machine-based computer-aided diagnosis system for Alzheimer's with visual support. *International Journal of Neural Systems*, 27(03):1650050, 2017.

[132] N.N. Kulkarni and V.K. Bairagi. Extracting salient features for EEG-based diagnosis of Alzheimer's disease using support vector machine classifier. *IETE Journal of Research*, 63(1):11–22, 2017.

[133] Ali Mazaheri, Katrien Segaert, John Olichney, Jin-Chen Yang, Yu-Qiong Niu, Kim Shapiro, and Howard Bowman. EEG oscillations during word processing predict mci conversion to Alzheimer's disease. *NeuroImage: Clinical*, 17:188–197, 2018.

[134] Gábor Gosztolya, Veronika Vincze, László Tóth, Magdolna Pákáski, János Kálmán, and Ildikó Hoffmann. Identifying mild cognitive impairment and mild Alzheimer's disease based on spontaneous speech using ASR and linguistic features. *Computer Speech & Language*, 53:181–197, 2019.

[135] Stefan Klöppel, Cynthia M. Stonnington, Carlton Chu, Bogdan Draganski, Rachael I. Scahill, Jonathan D. Rohrer, Nick C. Fox, Clifford R. Jack Jr, John Ashburner, and Richard S.J. Frackowiak. Automatic classification of mr scans in Alzheimer's disease. *Brain*, 131(3):681–689, 2008.

[136] Yudong Zhang, Zhengchao Dong, Preetha Phillips, Shuihua Wang, Genlin Ji, Jiquan Yang, and Ti-Fei Yuan. Detection of subjects and brain regions related to Alzheimer's disease using 3D MRI scans based on eigenbrain and machine learning. *Frontiers in Computational Neuroscience*, 9:66, 2015.

[137] Salim Lahmiri and Amir Shmuel. Performance of machine learning methods applied to structural MRI and ADAS cognitive scores in diagnosing Alzheimer's disease. *Biomedical Signal Processing and Control*, 2018.

[138] Jialin Peng, Xiaofeng Zhu, Ye Wang, Le An, and Dinggang Shen. Structured sparsity regularized multiple kernel learning for Alzheimer's disease diagnosis. *Pattern Recognition*, 88:370–382, 2019.

[139] Jonathan Stoeckel and Glenn Fung. SVM feature selection for classification of spect images of Alzheimer's disease using spatial information. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[140] Rémi Cuingnet, Joan Alexis Glaunès, Marie Chupin, Habib Benali, and Olivier Colliot. Spatial and anatomical regularization of SVM: a general framework for neuroimaging data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):682–696, 2013.

[141] Glenn Fung and Jonathan Stoeckel. SVM feature selection for classification of spect images of Alzheimer's disease using spatial information. *Knowledge and Information Systems*, 11(2):243–258, 2007.

[142] Yudong Zhang and Shuihua Wang. Detection of Alzheimer's disease by displacement field and machine learning. *PeerJ*, 3:e1251, 2015.

[143] Yitian Xu, Xianli Pan, Zhijian Zhou, Zhiji Yang, and Yuqun Zhang. Structural least square twin support vector machine for classification. *Applied Intelligence*, 42(3):527–536, 2015.

[144] Shen Lu, Yong Xia, Weidong Cai, Michael Fulham, David Dagan Feng, and Alzheimer's Disease Neuroimaging Initiative. Early identification of mild cognitive impairment using incomplete random forest-robust support vector machine and FDG-PET imaging. *Computerized Medical Imaging and Graphics*, 60:35–41, 2017.

[145] Saruar Alam, Goo-Rak Kwon, Ji-In Kim, and Chun-Su Park. Twin SVM-based classification of Alzheimer's disease using complex dual-tree wavelet principal coefficients and LDA. *Journal of Healthcare Engineering*, 2017, 2017.

[146] Zhuo Sun, Yuchuan Qiao, Boudewijn P.F. Lelieveldt, Marius Staring, and Alzheimer's Disease Neuroimaging Initiative. Integrating spatial-anatomical regularization and structure sparsity into SVM: Improving interpretation of Alzheimer's disease classification. *NeuroImage*, 178:445–460, 2018.

[147] Nianyin Zeng, Hong Qiu, Zidong Wang, Weibo Liu, Hong Zhang, and Yurong Li. A new switching-delayed-pso-based optimized SVM algorithm for diagnosis of Alzheimer's disease. *Neurocomputing*, 320:195–202, 2018.

[148] Xia-an Bi, Qing Shu, Qi Sun, and Qian Xu. Random support vector machine cluster analysis of resting-state fMRI in Alzheimer's disease. *PloS One*, 13(3):e0194479, 2018.

[149] I.A. Illán, J.M. Górriz, M.M. López, Javier Ramírez, Diego Salas-Gonzalez, Fermín Segovia, Rosa Chaves, and Carlos García Puntonet. Computer aided diagnosis of Alzheimer's disease using component based SVM. *Applied Soft Computing*, 11(2):2376–2382, 2011.

[150] Diego Salas-Gonzalez, Juan Manuel Górriz, Javier Ramírez, Míriam López, Ignacio Álvarez, Fermín Segovia, and Carlos García Puntonet. Computer aided diagnosis of Alzheimer's disease using support vector machines and classification trees. In *International Conference on Neural Information Processing*, pages 418–425. Springer, 2008.

[151] Javier Ramírez, J.M. Górriz, Diego Salas-Gonzalez, A. Romero, Míriam López, Ignacio Álvarez, and Manuel Gómez-Río. Computer-aided diagnosis of Alzheimer's type dementia combining support vector machines and discriminant set of features. *Information Sciences*, 237:59–72, 2013.

[152] Christiane Möller, Yolande A.L. Pijnenburg, Wiesje M. van der Flier, et al. Alzheimer disease and behavioral variant frontotemporal dementia: automatic classification based on cortical atrophy for single-subject diagnosis. *Radiology*, 279(3):838–848, 2015.

[153] Xiaojing Long, Lifang Chen, Chunxiang Jiang, Lijuan Zhang, and Alzheimer's Disease Neuroimaging Initiative. Prediction and classification of Alzheimer disease based on quantification of MRI deformation. *PloS One*, 12(3):e0173372, 2017.

[154] José María Mateos-Pérez, Mahsa Dadar, María Lacalle-Aurioles, Yasser Iturria-Medina, Yashar Zeighami, and Alan C Evans. Structural neuroimaging as clinical predictor: A review of machine learning applications. *NeuroImage: Clinical*, 2018.

[155] Xiaobo Chen, Jian Yang, Qiaolin Ye, and Jun Liang. Recursive projection twin support vector machine via within-class variance minimization. *Pattern Recognition*, 44(10-11):2643–2655, 2011.

[156] Xinjun Peng and De Chen. PTSVRs: Regression models via projection twin support vector machine. *Information Sciences*, 435:1–14, 2018.

[157] N.N. Zhao, X.Y. Ouyang, C. Gao, and Li Wang. A v-twin projection SVR with automatic accuracy adjustment. *Artificial Intelligence Review*, pages 1–17, 2019.

[158] Yuan-Hai Shao, Nai-Yang Deng, and Zhi-Min Yang. Least squares recursive projection twin support vector machine for classification. *Pattern Recognition*, 45(6):2299–2307, 2012.

[159] Zhi-Min Yang, He-Ji Wu, Chun-Na Li, and Yuan-Hai Shao. Least squares recursive projection twin support vector machine for multi-class classification. *International Journal of Machine Learning and Cybernetics*, 7(3):411–426, 2016.

[160] Shangjun Ma, Bo Cheng, Zhaowei Shang, and Geng Liu. Scattering transform and LSPTSVM based fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing*, 104:155–170, 2018.

[161] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[162] James C. Bezdek, Robert Ehrlich, and William Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[163] Paul S. Bradley and Olvi L. Mangasarian. K-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.

[164] Yuan-Hai Shao, Lan Bai, Zhen Wang, Xiang-Yu Hua, and Nai-Yang Deng. Proximal plane clustering via eigenvalues. *Procedia Computer Science*, 17:41–47, 2013.

[165] Reshma Rastogi and Aman Pal. Fuzzy semi-supervised weighted linear loss twin support vector clustering. *Knowledge-Based Systems*, 165:132–148, 2019.

[166] Zhen Wang, Yuan-Hai Shao, Lan Bai, and Nai-Yang Deng. Twin support vector machine for clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2583–2588, 2015.

[167] Lan Bai, Yuan-Hai Shao, Zhen Wang, and Chun-Na Li. Clustering by twin support vector machine and least square twin support vector classifier with uniform output coding. *Knowledge-Based Systems*, 163:227–240, 2019.

[168] Reshma Khemchandani, Aman Pal, and Suresh Chandra. Fuzzy least squares twin support vector clustering. *Neural Computing and Applications*, 29(2):553–563, 2018.

[169] Pak-Ming Cheung and James T. Kwok. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 193–200. ACM, 2006.

[170] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[171] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco 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.

[172] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[173] James M. Keller and Douglas J. Hunt. Incorporating fuzzy membership functions into the perceptron algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):693–699, 1985.

[174] Su-Gen Chen and Xiao-Jun Wu. A new fuzzy twin support vector machine for pattern classification. *International Journal of Machine Learning and Cybernetics*, 9(9):1553–1564, 2018.

[175] Kai Li and Hongyan Ma. A fuzzy twin support vector machine algorithm. *International Journal of Application or Innovation in Engineering and Management (IJAIEM)*, 2(3):459–465, 2013.

[176] Bin-Bin Gao, Jian-Jun Wang, Yao Wang, and Chan-Yun Yang. Coordinate descent fuzzy twin support vector machine for classification. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 7–12. IEEE, 2015.

[177] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. Learning from imbalanced data in presence of noisy and borderline examples. In *International Conference on Rough Sets and Current Trends in Computing*, pages 158–167. Springer, 2010.

[178] Yuan-Hai Shao, Chun-Hua Zhang, Xiao-Bo Wang, and Nai-Yang Deng. Improvements on twin support vector machines. *IEEE Transactions on Neural Networks*, 22(6):962–968, 2011.

[179] Yuchun Tang, Yan-Qing Zhang, Nitesh V. Chawla, and Sven Krasser. SVMs modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288, 2008.

[180] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[181] Dong-Jun Yu, Jun Hu, Zhen-Min Tang, Hong-Bin Shen, Jian Yang, and Jing-Yu Yang. Improving protein-atp binding residues prediction by boosting SVMs with random under-sampling. *Neurocomputing*, 104:180–190, 2013.

[182] Mittul Singh, Jivitej Chadha, Puneet Ahuja, Jayadeva, and Suresh Chandra. Reduced twin support vector regression. *Neurocomputing*, 74(9):1474–1477, 2011.

[183] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[184] Yitian Xu. Maximum margin of twin spheres support vector machine for imbalanced data classification. *IEEE Transactions on Cybernetics*, 47(6):1540–1550, 2016.

[185] Xinjun Peng. A ν-twin support vector machine (ν-TSVM) classifier and its geometric algorithms. *Information Sciences*, 180(20):3863–3875, 2010.

[186] D.R. Musicant. NDC: normally distributed clustered datasets. *Computer Sciences Department, University of Wisconsin, Madison*, 1998.

[187] Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231, 2019.

[188] Xue Bai and Vladimir Cherkassky. Gender classification of human faces using inference through contradictions. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 746–750. IEEE, 2008.

[189] Shuo Chen and Changshui Zhang. Image classification via SVM using in-between universum samples. In *16th IEEE International Conference on Image Processing (ICIP)*, pages 1421–1424, 2009.

[190] Vladimir Cherkassky and Wuyang Dai. Empirical study of the universum SVM learning for high-dimensional data. In *International Conference on Artificial Neural Networks*, pages 932–941. Springer, 2009.

[191] Vladimir Cherkassky, Sauptik Dhar, and Wuyang Dai. Practical conditions for effectiveness of the universum learning. *IEEE Transactions on Neural Networks*, 22(8):1241–1255, 2011.

[192] Ralph G. Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

[193] Philipp Wagner. Face recognition with GNU octave/MATLAB, 2012.

[194] Marian Stewart Bartlett, Javier R. Movellan, and Terrence J. Sejnowski. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6):1450–1464, 2002.

[195] Ivor W. Tsang, Andras Kocsor, and James T. Kwok. Efficient kernel feature extraction for massive data sets. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 724–729, 2006.

[196] Chunfeng Lian, Mingxia Liu, Jun Zhang, and Dinggang Shen. Hierarchical fully convolutional network for joint atrophy localization and Alzheimer's disease diagnosis using structural MRI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[197] Raymond Y. Lo and William J. Jagust. Predicting missing biomarker data in a longitudinal study of Alzheimer disease. *Neurology*, 78(18):1376–1382, 2012.

[198] John Ashburner. A fast diffeomorphic image registration algorithm. *NeuroImage*, 38(1):95–113, 2007.

[199] S.S. Keller, U.C. Wieshmann, C.E. Mackay, C.E. Denby, J. Webb, and N. Roberts. Voxel based morphometry of grey matter abnormalities in patients with medically intractable temporal lobe epilepsy: Effects of side of seizure onset and epilepsy duration. *Journal of Neurology, Neurosurgery & Psychiatry*, 73(6):648–655, 2002.

[200] Liam M. O'Brien, David A. Ziegler, Curtis K. Deutsch, Jean A. Frazier, Martha R. Herbert, and Joseph J. Locascio. Statistical adjustments for brain

size in volumetric neuroimaging studies: Some practical implications in methods. *Psychiatry Research: Neuroimaging*, 193(2):113–122, 2011.

[201] Zhe Xiao, Yi Ding, Tian Lan, Cong Zhang, Chuanji Luo, and Zhiguang Qin. Brain MR image classification for Alzheimer's disease diagnosis based on multifeature fusion. *Computational and Mathematical Methods in Medicine*, 2017, 2017.

[202] Vahab Youssofzadeh, Bernadette McGuinness, Liam P. Maguire, and KongFatt Wong-Lin. Multi-kernel learning with dartel improves combined mri-pet classification of Alzheimer's disease in AIBL data: group and individual analyses. *Frontiers in Human Neuroscience*, 11:380, 2017.

[203] Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with various feature selection strategies. In *Feature Extraction*, pages 315–324. Springer, 2006.

[204] Eric Westman, J-Sebastian Muehlboeck, and Andrew Simmons. Combining MRI and CSF measures for classification of Alzheimer's disease and prediction of mild cognitive impairment conversion. *NeuroImage*, 62(1):229–238, 2012.

[205] Martin Reuter, Nicholas J. Schmansky, H. Diana Rosas, and Bruce Fischl. Within-subject template estimation for unbiased longitudinal image analysis. *NeuroImage*, 61(4):1402–1418, 2012.

[206] Lauge Sørensen, Christian Igel, Akshay Pai, Ioana Balas, Cecilie Anker, Martin Lillholm, Mads Nielsen, and Alzheimer's Disease Neuroimaging Initiative. Differential diagnosis of mild cognitive impairment and Alzheimer's disease using structural MRI cortical thickness, hippocampal shape, hippocampal texture, and volumetry. *NeuroImage: Clinical*, 13:470–482, 2017.

[207] Mei-Ling Huang, Yung-Hsiang Hung, W.M. Lee, R.K. Li, and Bo-Ru Jiang. SVM-RFE based feature selection and Taguchi parameters optimization for multiclass SVM classifier. *The Scientific World Journal*, 2014, 2014.

[208] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *NeuroImage*, 31(3):1116–1128, 2006.

[209] James Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. *The Visualization Handbook*, 717, 2005.

[210] Joseph A. Maldjian, Paul J. Laurienti, Robert A. Kraft, and Jonathan H. Burdette. An automated method for neuroanatomic and cytoarchitectonic atlas-based interrogation of fMRI data sets. *NeuroImage*, 19(3):1233–1239, 2003.

[211] Carlton Chu, Ai-Ling Hsu, Kun-Hsien Chou, Peter Bandettini, ChingPo Lin, and Alzheimer's Disease Neuroimaging Initiative. Does feature selection improve classification accuracy? Impact of sample size and feature selection on classification using anatomical magnetic resonance images. *NeuroImage*, 60(1):59–70, 2012.

[212] Sabine Krumm, Sasa L. Kivisaari, Alphonse Probst, Andreas U. Monsch, Julia Reinhardt, Stephan Ulmer, Christoph Stippich, Reto W. Kressig, and Kirsten I. Taylor. Cortical thinning of parahippocampal subregions in very early Alzheimer's disease. *Neurobiology of Aging*, 38:188–196, 2016.

[213] Yong Fan, Nematollah Batmanghelich, Chris M. Clark, Christos Davatzikos, and Alzheimer's Disease Neuroimaging Initiative. Spatial patterns of brain atrophy in mci patients, identified via high-dimensional pattern classification, predict subsequent cognitive decline. *NeuroImage*, 39(4):1731–1743, 2008.

[214] Sabine Derflinger, Christian Sorg, Christian Gaser, Nicholas Myers, Milan Arsic, Alexander Kurz, Claus Zimmer, Afra Wohlschläger, and Mark Mühlau. Grey-matter atrophy in Alzheimer's disease is asymmetric but not lateralized. *Journal of Alzheimer's Disease*, 25(2):347–357, 2011.

[215] O. Bugiani, J. Constantinidis, B. Ghetti, C. Bouras, and F. Tagliavini. Asymmetrical cerebral atrophy in Alzheimer's disease. *Clinical Neuropathology*, 10(2):55–60, 1991.

[216] Philip S.J. Weston, Ivor J.A. Simpson, Natalie S. Ryan, Sebastien Ourselin, and Nick C. Fox. Diffusion imaging changes in grey matter in Alzheimer's disease: A potential marker of early neurodegeneration. *Alzheimer's Research & Therapy*, 7(1):47, 2015.

[217] Stéphane P. Poulin, Rebecca Dautoff, John C. Morris, Lisa Feldman Barrett, Bradford C. Dickerson, and Alzheimer's Disease Neuroimaging Initiative. Amygdala atrophy is prominent in early Alzheimer's disease and relates to symptom severity. *Psychiatry Research: Neuroimaging*, 194(1):7–13, 2011.

[218] A.T. Du, N. Schuff, D. Amend, et al. Magnetic resonance imaging of the entorhinal cortex and hippocampus in mild cognitive impairment and Alzheimer's disease. *Journal of Neurology, Neurosurgery & Psychiatry*, 71(4):441–447, 2001.

[219] Latha Velayudhan, Petroula Proitsi, Eric Westman, et al. Entorhinal cortex thickness predicts cognitive decline in Alzheimer's disease. *Journal of Alzheimer's Disease*, 33(3):755–766, 2013.

[220] Daniel H. Adler, Laura E. M. Wisse, Ranjit Ittyerah, et al. Characterizing the human hippocampus in aging and Alzheimer's disease using a computational atlas derived from ex vivo MRI and histology. *Proceedings of the National Academy of Sciences*, 115(16):4252–4257, 2018.

[221] Bradford C. Dickerson and Reisa A. Sperling. Functional abnormalities of the medial temporal lobe memory system in mild cognitive impairment and Alzheimer's disease: Insights from functional MRI studies. *Neuropsychologia*, 46(6):1624–1635, 2008.

[222] Gary W. Van Hoesen, Jean C. Augustinack, Jason Dierking, Sarah J. Redman, and Ramasamy Thangavel. The parahippocampal gyrus in Alzheimer's disease:

clinical and preclinical neuroanatomical correlates. *Annals of the New York Academy of Sciences*, 911(1):254–274, 2000.

[223] F.A. Spanhol, L.S. Oliveira, C. Petitjean, and L. Heutte. A dataset for breast cancer histopathological image classification. *IEEE Transactions on Biomedical Engineering*, 63(7):1455–1462, 2015.

[224] Chandan Gautam, Pratik K. Mishra, Aruna Tiwari, Bharat Richhariya, Hari Mohan Pandey, Shuihua Wang, M. Tanveer, and Alzheimer's Disease Neuroimaging Initiative. Minimum variance-embedded deep kernel regularized least squares method for one-class classification and its applications to biomedical data. *Neural Networks*, 123:191–216, 2020.

[225] Reshma Khemchandani, Pooja Saigal, and Suresh Chandra. Angle-based twin support vector machine. *Annals of Operations Research*, 269(1-2):387–417, 2018.

[226] Ling-Wei Huang, Yuan-Hai Shao, Jun Zhang, Yu-Ting Zhao, and Jia-Ying Teng. Robust rescaled hinge loss twin support vector machine for imbalanced noisy classification. *IEEE Access*, 7:65390–65404, 2019.

[227] Yuan-Hai Shao, Zhen Wang, Wei-Jie Chen, and Nai-Yang Deng. Least squares twin parametric-margin support vector machine for classification. *Applied Intelligence*, 39(3):451–464, 2013.

[228] Alan L. Yuille and Anand Rangarajan. The concave-convex procedure (CCCP). In *Advances in Neural Information Processing Systems*, pages 1033–1040, 2002.

[229] Michael Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba. Coding facial expressions with Gabor wavelets. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205. IEEE, 1998.

[230] Ling Huang, Donghui Yan, Nina Taft, and Michael I. Jordan. Spectral clustering with perturbed data. In *Advances in Neural Information Processing Systems*, pages 705–712, 2009.

[231] Huihui Li and Guihua Wen. Sample awareness-based personalized facial expression recognition. *Applied Intelligence*, pages 1–14, 2019.

[232] Richard J. Harris, Andrew W. Young, and Timothy J. Andrews. Brain regions involved in processing facial identity and expression are differentially selective for surface and edge information. *NeuroImage*, 97:217–223, 2014.

[233] Ramesh Kumar Lama, Jeonghwan Gwak, Jeong-Seon Park, and Sang-Woong Lee. Diagnosis of Alzheimer's disease based on structural MRI images using a regularized extreme learning machine and PCA features. *Journal of Healthcare Engineering*, 2017, 2017.

[234] Zhe Wang, Yu Zheng, David C. Zhu, Andrea C. Bozoki, and Tongtong Li. Classification of Alzheimer's disease, mild cognitive impairment and normal control subjects using resting-state fMRI based network connectivity analysis. *IEEE Journal of Translational Engineering in Health and Medicine*, 6:1–9, 2018.